Lecture Notes on Probabilistic Concurrency

Yuxin Deng

June 11, 2008

Preface

These are lecture notes I drafted for a semantics course (Spring 2008) at Shanghai Jiao Tong University for graduate students interested in formal methods. They are not in a polished form and there may be some errors and typos. The citation of the references is also biased by my own viewpoint; I apologize for any inaccuracy or omission. The contents are updated from time to time. The latest version is available from the following link:

`http://basics.sjtu.edu.cn/~yuxin`

I should very much appreciate being told of any corrections or possible improvements. My e-mail address is `deng-yx@cs.sjtu.edu.cn`.

Shanghai, China Yuxin Deng

Contents

Chapter 1

Mathematical Preliminaries

1.1 Lattice theory

Definition 1.1 *A set X with a binary relation $\sqsubseteq$ is called a* partially ordered set *if the following holds for all $x, y, z \in X$:*

1. *$x \sqsubseteq x$ (Reflexivity)*
2. *if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$ (Antisymmetry)*
3. *if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$ (Transitivity)*

Below are some basic concepts from order theory. An element $x \in X$ is called an *upper bound* for a subset $Y \sqsubseteq X$, if $y \sqsubseteq x$ for all $y \in Y$. Dually, x is an *lower bound* for Y, if $x \sqsubseteq y$ for all $y \in Y$. If $y \in Y$ is an upper bound for Y, then y is said to be the *largest element*. We can dually define the *least element*. In the presence of a least element we speak of a *pointed* partially ordered set. If the set of upper bounds for Y has a least element x, then x is called the *supremum* or *join* of Y, written $\bigsqcup Y$. Dually we have *infimum* or *meet* and write $\bigsqcap Y$. We call X a *complete lattice* if suprema and infima exist for all the subsets of X.

Example 1.2 *Given a set X, its powerset $\mathcal{P}(X) = \{Y \mid Y \subseteq X\}$ with the inclusion relation $\subseteq$ forms a complete lattice whose join and meet being set union and intersection, respectively.*

Given a function $f : X \to Y$ and a set $Z \subseteq X$, we write $f(Z)$ for the set $\{f(z) \mid z \in Z\}$. Given a partially ordered set X and a function $f : X \to X$, we say $x \in X$ a *fixed point* (resp. *pre-fixed point*, *post-fixed point*) of f if $x = f(x)$ (resp. $f(x) \sqsubseteq x$, $x \sqsubseteq f(x)$).

Definition 1.3 *Let X and Y be partially ordered sets. A function $f : X \to Y$ is called* monotone *if for all $x, y \in X$ with $x \sqsubseteq y$ it holds that $f(x) \sqsubseteq f(y)$ in Y.*

Theorem 1.4 (Knaster-Tarski fixed point theorem) *If X is a complete lattice then every monotone function f from X to X has a fixed point. The least of these is given by*

$$lfp(f) = \bigsqcap \{x \in X \mid f(x) \sqsubseteq x\},$$

the greatest by

$$gfp(f) = \bigsqcup \{x \in X \mid x \sqsubseteq f(x)\}.$$

Proof: Let $Y = \{x \in X \mid f(x) \sqsubseteq x\}$ and $y = \bigsqcap Y$. For each $x \in Y$ we have $y \sqsubseteq x$ and $f(y) \sqsubseteq f(x) \sqsubseteq x$. Taking the infimum we get $f(y) \sqsubseteq \bigsqcap f(Y) \sqsubseteq \bigsqcap Y = y$, thus $y \in Y$. On the other hand, $x \in Y$ implies $f(x) \in Y$ by monotonicity. Applying this to y yields $f(y) \in Y$ which implies $y \sqsubseteq f(y)$. □

Usually, an object is defined *inductively* (resp. *coinductively*) if it is the *least* (resp. *greatest*) fixed point of a function. So the above theorem provides two proof principles: the *induction principle* says that to show $lfp(f) \sqsubseteq x$ it is enough to prove $f(x) \sqsubseteq x$; the *coinduction principle* says that to show $x \sqsubseteq gfp(f)$ it is enough to prove $x \sqsubseteq f(x)$.

Definition 1.5 *Given a complete lattice X, the function $f : X \to X$ is* continuous *if it preserves increasing chains, i.e. for all sequences $x_0 \leq x_1 \leq ...$ we have $f(\bigsqcup_{n\geq 0} x_n) = \bigsqcup_{n\geq 0} f(x_n)$. Dually, f is* cocontinuous *if it preserves decreasing chains.*

Notice that continuity and cocontinuity imply monotonicity. For example, if f is continuous and $x \sqsubseteq y$, then we obtain from the increasing sequence $x \sqsubseteq y \sqsubseteq y \sqsubseteq ...$ that $f(\bigsqcup\{x, y\}) = f(y) = \bigsqcup\{f(x), f(y)\}$, which means $f(x) \sqsubseteq f(y)$. With continuity and cocontinuity we can construct in a tractable way the least and greatest fixed point, respectively.

Proposition 1.6 *Let X be a complete lattice.*

1. *Every continuous function f on X has a least fixed point, given by $\bigsqcup_{n\geq 0} f^n(\bot)$, where $\bot$ is the bottom element of the lattice, and $f^n(\bot)$ is the n-th iteration of f on $\bot$: $f^0(\bot) := \bot$ and $f^{n+1}(\bot) := f(f^n(\bot))$ for $n \geq 0$.*
2. *Every cocontinuous function f on X has a greatest fixed point, given by $\bigsqcap_{n\geq 0} f^n(\top)$, where $\top$ is the top element of the lattice.*

Proof: We only prove the first clause, since the second one is dual.

We notice that $\bot \sqsubseteq f(\bot)$ and then monotonicity of f yields an increasing sequence:

$$\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq ...$$

By continuity of f we have $f(\bigsqcup_{n\geq 0} f^n(\bot)) = \bigsqcup_{n\geq 0} f^{n+1}(\bot)$ and the latter is equal to $\bigsqcup_{n\geq 0} f^n(\bot)$.

If x is also a fixed point of f, then we have $\bot \sqsubseteq x$ and then $f^n(\bot) \sqsubseteq x$ for all n by induction. So x is an upper bound of all $f^n(\bot)$. □

1.2 Domain theory

Definition 1.7 *Let X be a partially ordered set. A subset Y of X is* directed*, if it is non-empty and each pair of elements of Y has an upper bound in Y.*

Simple examples of directed sets are chains, which are totally ordered non-empty subsets, i.e. for each pair x, y either $x \sqsubseteq y$ or $y \sqsubseteq x$ holds.

Definition 1.8 *A partially ordered set X is* directed-complete*, or called DCPO, if every directed subset has a supremum in it.*

Every complete lattice is also a DCPO. Every finite partially ordered set is a DCPO.

Proposition 1.9 *A partially ordered set X is a DCPO if and only if each chain in X has a supremum.*

Proof: The proof uses the Axiom of Choice, and can be found in [Mar76]. □

Definition 1.10 *Let X and Y be DCPO's. A function $X \to Y$ is* (Scott-) continuous *if it is monotone and if for each directed subset Z of X we have $f(\bigsqcup Z) = \bigsqcup f(Z)$. We denote the set of all continuous functions from X to Y, ordered pointwise, by $[X \to Y]$.*

Proposition 1.11 *Let X and Y be DCPO's. Then $[X \to Y]$ is also a DCPO. Directed suprema in $[X \to Y]$ are calculated pointwise.*

Proof: Let $\mathcal{F}$ be a directed subset $[X \to Y]$. It follows that for any $x \in X$, the set $\{f(x) \mid f \in \mathcal{F}\}$ is directed. Let $f^* : X \to Y$ be the function defined by $f^*(x) := \bigsqcup_{f \in \mathcal{F}} f(x)$. We now show that $f^* \in [X \to Y]$. Let $Z \subseteq X$ be directed.

$$\begin{array}{rcl} f^*(\bigsqcup Z) & = & \bigsqcup_{f \in \mathcal{F}} f(\bigsqcup Z) \\ & = & \bigsqcup_{f \in \mathcal{F}} \bigsqcup_{x \in Z} f(x) \\ & = & \bigsqcup_{x \in Z} \bigsqcup_{f \in \mathcal{F}} f(x) \\ & = & \bigsqcup_{x \in Z} f^*(x) \end{array}$$

Therefore, f^* is continuous. □

In DCPO we have a counterpart of Proposition 1.6 (1).

Proposition 1.12 *Let X be a pointed DCPO. Every continuous function f on X has a least fixed point. It is given by $\bigsqcup_{n \in \mathbb{N}} f^n(\bot)$.*

Proof: Similar to the proof of Proposition 1.6 (1). □

Let $\mathbb{R}^+$ be the set of non-negative real numbers. The following property is very useful. It says that the directed sup and countable sum can be interchanged.

Lemma 1.13 *Let S be a set and $(S \to \mathbb{R}^+)$ be the set of all functions from S to $\mathbb{R}^+$. Suppose $\{f_i \mid i \in I\}$ is any directed subset of $(S \to \mathbb{R}^+)$.*

1. *If S is finite, then*
$$\sum_{s \in S} \bigsqcup_{i \in I} f_i(s) \;=\; \bigsqcup_{i \in I} \sum_{s \in S} f_i(s).$$
2. *If S is countable and the partial sum $S_n := \sum_{j=1}^{n} \bigsqcup_{i \in I} f_i(s_j)$ is bounded, i.e. there exists some $c \in \mathbb{R}^+$ such that $S_n \leq c$ for any n, then*
$$\sum_{s \in S} \bigsqcup_{i \in I} f_i(s) \;=\; \bigsqcup_{i \in I} \sum_{s \in S} f_i(s).$$

Proof:

1. Since S is finite, we can assume that $|S| = N$ for some $N \in \mathbb{N}$. Let ϵ be any positive real number. For each $s \in S$, there is some index i_s such that $0 \leq \bigsqcup_{i \in I} f_i(s) - f_i(s) \leq \frac{\epsilon}{N}$ for all $i > i_s$. Let $i_S = max\{i_s \mid s \in S\}$. For any $s \in S$, we have $0 \leq \bigsqcup_{i \in I} f_i(s) - f_i(s) \leq \frac{\epsilon}{N}$ for all $i > i_S$. Summing up over all $s \in S$, we get $0 \leq \sum_{s \in S} \bigsqcup_{i \in I} f_i(s) - \sum_{s \in S} f_i(s) \leq \epsilon$ for all $i > i_S$. Therefore, $\bigsqcup_{i \in I} \sum_{s \in S} f_i(s) = \lim_{i \to \infty} \sum_{s \in S} f_i(s) = \sum_{s \in S} \bigsqcup_{i \in I} f_i(s)$.
2. Since the sequence $\{S_n\}_{n \in \mathbb{N}}$ is increasing and bounded, it converges to $\sum_{s \in S} \bigsqcup_{i \in I} f_i(s)$. Let ϵ be any positive real number. We can take a finite subset S' of S which is large enough so that
$$0 \leq \sum_{s \in S} \bigsqcup_{i \in I} f_i(s) - \sum_{s \in S'} \bigsqcup_{i \in I} f_i(s) \leq \frac{\epsilon}{2}. \tag{1.1}$$
With the same argument as in the proof the first clause, we can choose an index $i_{S'}$ so that
$$0 \leq \sum_{s \in S'} \bigsqcup_{i \in I} f_i(s) - \sum_{s \in S'} f_i(s) \leq \frac{\epsilon}{2} \tag{1.2}$$
for all $i > i_{S'}$. We observe that $f_i(s) \leq \bigsqcup_{i \in I} f_i(s)$, so the sequence $\{\sum_{j=1}^{n} f_i(s)\}_{n \in \mathbb{N}}$, for any $i \in I$, is increasing and bounded, thus converges to $\sum_{s \in S} f_i(s)$. Therefore, there exists some $N \in \mathbb{N}$ such that
$$0 \leq \sum_{s \in S} f_i(s) - \sum_{j=1}^{N} f_i(s) \leq \frac{\epsilon}{2} \tag{1.3}$$

for all $i \in I$. Without loss of generality, we assume that $\{s_1, ..., s_N\} \subseteq S'$. It follows from (1.3) that

$$-\frac{\epsilon}{2} \leq \sum_{s \in S'} f_i(s) - \sum_{s \in S} f_i(s) \leq 0 \tag{1.4}$$

for all $i \in I$. Take the sum of the three inequalities (1.1), (1.2) and (1.4), we obtain

$$-\frac{\epsilon}{2} \leq \sum_{s \in S} \bigsqcup_{i \in I} f_i(s) - \sum_{s \in S} f_i(s) \leq \epsilon \tag{1.5}$$

for all $i > i_{S'}$. Therefore, $\bigsqcup_{i \in I} \sum_{s \in S} f_i(s) = \lim_{i \to \infty} \sum_{s \in S} f_i(s) = \sum_{s \in S} \bigsqcup_{i \in I} f_i(s)$.

□

1.3 Metric spaces

Definition 1.14 *A* metric space *is a pair (X, d) consisting of a set X and a distance function $m : X \times X \longrightarrow \mathbb{R}^+$ satisfying*

1. *for all $x, y \in X$, $d(x, y) = 0$ iff $x = y$ (isolation)*
2. *for all $x, y \in X$, $d(x, y) = d(y, x)$ (symmetry)*
3. *for all $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).*

If we replace the first clause with $\forall x \in X : d(x, x) = 0$, we obtain the definition of pseudometric space.

A metric d is c-bounded if $\forall x, y \in X : d(x, y) \leq c$, where c is a positive real number.

Example 1.15 *Let X be a set. The discrete metric $m : X \times X \longrightarrow [0, 1]$ is defined by*

$$d(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise.} \end{cases}$$

Definition 1.16 *A sequence (x_n) in a metric space (X, d) is* convergent *to $x \in X$, if for an arbitrary $\epsilon > 0$ there exists $N \in \mathbb{N}$ such that $d(x_n, x) < \epsilon$ whenever $n > N$.*

Definition 1.17 *A sequence (x_n) in a metric space (X, d) is called a* Cauchy sequence *if for an arbitrary $\epsilon > 0$ there exists $N \in \mathbb{N}$ such that $d(x_m, x_n) < \epsilon$ whenever $m, n > N$.*

Definition 1.18 *A metric space is* complete *if every Cauchy sequence is convergent.*

For example, the space of real numbers with the usual metric is complete.

Example 1.19 *Let X be a non-empty set and F denote the collection of functions from X to the interval $[0, 1]$. A metric is defined on F as follows:*

$$d(f, g) \ := \ sup_{x \in X} |f(x) - g(x)|$$

In fact, (F, d) is a complete metric space. Let (f_n) be a Cauchy sequence in F. Then for every $x \in X$, the sequence $(f_n(x))$ is Cauchy; and since $[0, 1]$ is complete, the sequence converges to some $a_x \in [0, 1]$. Let f be the function defined by $f(x) = a_x$. Thus (f_n) converges to f.

Definition 1.20 *Let (X, d) be a metric space. A function $f : X \to X$ is said to be a* contraction mapping *if there is a constant δ with $0 \leq \delta < 1$ such that*

$$d(f(x), f(y)) \leq \delta \cdot d(x, y)$$

for all $x, y \in X$.

Contractions have an important property.

Theorem 1.21 (Banach fixed point theorem) *Every contraction on a complete metric space has a unique fixed point.*

Proof: Let (X, d) be a complete metric space, and f be a contraction mapping on (X, d) with constant δ. For any $x_0 \in X$, define the sequence (x_n) by $x_{n+1} := f(x_n)$ for $n \geq 0$. Let $a := d(x_0, x_1)$. It is easy to show that

$$d(x_n, x_{n+1}) \leq \delta^n \cdot a$$

by repeated application of the property $d(f(x), f(y)) \leq \delta \cdot d(x, y)$. Given any $\epsilon > 0$, it is possible to choose a natural number N such that $\frac{\delta^n}{1-\delta} a < \epsilon$ for all $n \geq N$. Now, for any $m, n \geq N$ with $m \leq n$,

$$\begin{aligned} d(x_m, x_n) &\leq d(x_m, x_{m+1}) + d(x_{m+1}, x_{m+2}) + ... + d(x_{n-1}, x_n) \\ &\leq \delta^m \cdot a + \delta^{m+1} \cdot a + ... + \delta^{n-1} \cdot a \\ &= \delta^m \tfrac{1-\delta^{n-m}}{1-\delta} a \\ &< \tfrac{\delta^m}{1-\delta} a < \epsilon \end{aligned}$$

by repeated application of the triangle inequality. So the sequence (x_n) is a Cauchy sequence. Since (X, d) is complete, the sequence has a limit in (X, d). We define x^* to be this limit and show that it is a fixed point of f. Suppose it is not, then $a^* := d(x^*, f(x^*)) > 0$. Since (x_n) converges to x^*, there exists some $N \in \mathbb{N}$ such that $d(x_n, x^*) < \frac{a^*}{2}$ for all $n \geq N$. Then

$$\begin{aligned} d(x^*, f(x^*)) &\leq d(x^*, x_{N+1}) + d(x_{N+1}, f(x^*)) \\ &\leq d(x^*, x_{N+1}) + \delta \cdot d(x_N, x^*) \\ &< \tfrac{a^*}{2} + \tfrac{a^*}{2} = a^*, \end{aligned}$$

this gives rise to a contradiction. So x^* is a fixed point of f. It is also unique. Otherwise, suppose there is another fixed point x'; we have $d(x', x^*) > 0$ since $x' \neq x^*$. But then we would have

$$d(x', x^*) = d(f(x'), f(x^*)) \leq \delta \cdot d(x', x^*) < d(x', x^*).$$

Therefore, x^* is the unique fixed point of f. □

1.4 Probability and measure theory

In this section, we recall some basic concepts from probaility and measure theory. More details can be found in many textbooks, for example [Bil95].

Definition 1.22 *Let X be an arbitrary non-empty set and $\mathcal{X}$ a family of subsets of X. We say that $\mathcal{X}$ is a* field *on X if*

1. *the emptyset $\emptyset \in \mathcal{X}$;*
2. *whenever $A \in \mathcal{X}$, then the complement $X \backslash A \in \mathcal{X}$;*
3. *whenever $A, B \in \mathcal{X}$, then $A \cup B \in \mathcal{X}$.*

A field $\mathcal{X}$ is a σ-algebra *if it is closed under countable union: whenever $A_i \in \mathcal{X}$ for $i \in \mathbb{N}$, then $\bigcup_{i \in \mathbb{N}} A_i \in \mathcal{X}$.*

The elements of a σ-algebra are called *measurable sets*, and $(X, \mathcal{X})$ is called a *measurable space*. A σ-algebra *generated* by a family of sets $\mathcal{X}$, denoted $\sigma(\mathcal{X})$, is the smallest σ-algebra that contains $\mathcal{X}$. The existence of $\sigma(\mathcal{X})$ is ensured by the following proposition.

Proposition 1.23 *For any non-empty set X and $\mathcal{X}$ a family of subsets of X, there exists a unique smallest σ-algebra containing $\mathcal{X}$.* □

Definition 1.24 *Let $(X, \mathcal{X})$ be a measurable space. A function $\mu : \mathcal{X} \to [0,1]$ is a* probability measure *on $(X, \mathcal{X})$ and $(X, \mathcal{X}, \mu)$ a* probability space*, if μ satisfies the following properties:*

1. $\mu(X) = 1$
2. $\mu(\bigcup_i A_i) = \sum_i \mu(A_i)$ *for any countable disjoint elements $A_1, A_2, ...$ of $\mathcal{X}$.*

The measure μ is also called a *probability distribution*, the set X a *sample space*, and the elements of $\mathcal{X}$ *events*.

Definition 1.25 *A family $\mathcal{X}$ of subsets of X is called a* semi-ring *if*

1. $\emptyset \in \mathcal{X}$*;*
2. *whenever $A, B \in \mathcal{X}$, then $A \cap B \in \mathcal{X}$;*
3. *if $A, B \in \mathcal{X}$ and $A \subseteq B$, then there are* finitely *many* pairwise disjoint *subsets $C_1, ..., C_k \in \mathcal{X}$ such that $B \backslash A = \bigcup_{i=1}^{k} C_i$.*

Theorem 1.26 *If $\mathcal{X}$ is a semi-ring on X and $\mu : \mathcal{X} \to [0, \infty]$ satisfies*

1. $\mu(\emptyset) = 0$
2. $\mu(\bigcup_{i=1}^{k} A_i) = \sum_{i=1}^{k} \mu(A_i)$ *for any finite disjoint elements $A_1, ., A_k \in \mathcal{X}$*
3. $\mu(\bigcup_i A_i) \leq \sum_i \mu(A_i)$ *for any countable elements $A_1, A_2, ... \in \mathcal{X}$,*

then μ extends to a unique measure on the σ-algebra generated by $\mathcal{X}$. □

See [Bil95] for a proof of this theorem.

Let $(X, \mathcal{X}, \mu)$ be a probability space. A function $f : X \to \mathbb{R}^+$ is said to be a random variable. The *expectation* or average value with respect to the measure μ is given by the following integral:

$$E[f] := \int_{x \in X} f(x) d\mu \ .$$

Chapter 2

Classical Process Algebras

A great amount of work in process algebra has centered around behavioural relations such as equivalences and refinement preorders as a basis for establishing system correctness. Usually both specifications and implementations are written as process terms in the same algebra, where a specification describes the expected high-level behaviour of the system under consideration and an implementation gives the detailed procedure of achieving the behaviour. An appropriate equivalence or preorder is then chosen to verify that the implementation conforms to the specification. Informally, for equivalence-based reasoning, a correct implementation has "the same behaviour" as that of the specification; for a preorder-based reasoning, a correct implementation has a behaviour "at least as good as" that required by the specification. A distinguished feature of such process-algebraic approaches of system verification is compositional reasoning in the sense that specifications and implementations can be built up from their components. In the last three decades, many different process algebras have been proposed, and a lot of equivalences and preorders have been developed to capture various aspects of behaviour.

Most of existing behavioural relations are defined on top of labelled transition systems, which offer a semantic model of systems, rather than on a particular syntax of process algebra. Equivalences and preorders defined in this way are applicable to any algebra with a semantics given by labelled transition systems. Moreover, for finitary labelled transition systems, these behavioural relations could be computed in a purely mechanical manner, so automatic verification tools may be developed to check that implementations conform to specifications.

2.1 Labelled transition systems

A *process* is the behaviour of a system such as a machine, a communication protocol, a game player etc. It is usually described by a labelled transition system, which is essentially a labelled directed graph.

Definition 2.1 (Labelled Transition Systems) *A* labelled transition system *(LTS) is a triple* $(S, Act, \rightarrow)$*, where*

1. S *is a set of states*
2. Act *is a set of actions*
3. $\longrightarrow \subseteq S \times Act \times S$ *is the transition relation.*

It is usual to use the more intuitive notation $s \xrightarrow{\alpha} s'$ *instead of* $(s, \alpha, s') \in \longrightarrow$.

The central topic in the study of processes is when should two processes be considered equal, i.e. two systems behave similarly. It is widely accepted that two processes should be equivalent if we cannot distinguish them by interacting with them. That is, the observations we could make

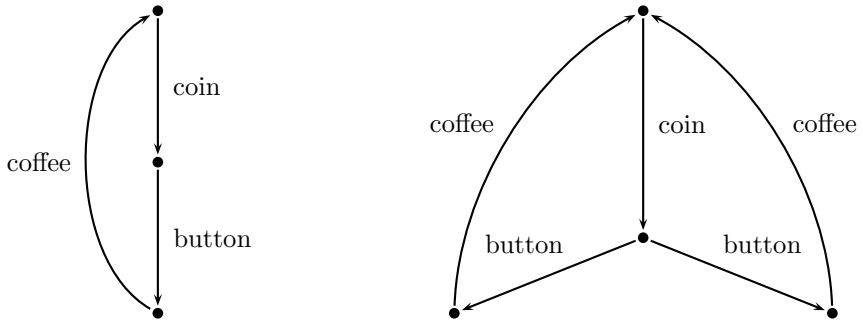


Figure 2.1: Two vending machines

while interacting with them are the same. Formally speaking, two processes can be identified if their *observational semantics* are the same.

Since processes can be described as labelled directed graphs, we think of the graph isomorphism from graph theory.

Definition 2.2 *A* graph isomorphism *between two LTSs* $(S_1, Act, \rightarrow)$ *and* $(S_2, Act, \rightarrow)$ *is a bijective function* $f : S_1 \rightarrow S_2$ *such that* $s \xrightarrow{\alpha} t$ *iff* $f(s) \xrightarrow{\alpha} f(t)$.

Example 2.3 *Imagine we have two vending machines. Machine A has a slot, a button, and a tray. Whenever a coin is inserted in the slot and the button is pushed, a cup of coffee is delivered in the tray. Machine B also has a slot and a tray but with two buttons. Whenever a coin is inserted in the slot and any of the two buttons is pushed, a cup of coffee is delivered in the tray. Their behaviours are described in Figure 2.1.*

Intuitively the two machines exhibit similar behaviour and should not be distinguished. However, there is no graph isomorphism between their LTSs.

This example shows that graph isomorphism is too strong to be a good *behavioural equivalence*. A natural alternative is *trace equivalence* from automata theory. For any alphabet Σ, let Σ^* be the set of finite sequences over Σ.

Definition 2.4 *A sequence of actions* $a_1...a_n \in Act^*$ *for some* $n > 0$ *is a* trace *of* $s \in S$ *if there are some states* $s_1, ..., s_n$ *such that* $s \xrightarrow{a} s_1 \xrightarrow{a} ... \xrightarrow{a} s_n$. *We say* s *and* t *are* trace equivalent, *written* $s =_T t$, *if they have the same set of traces.*

Example 2.5 *Consider a bad vending machine which is the same as the second machine in Example 2.3 except that one button does not function, i.e. when it is pushed the machine is deadlocked. Its behaviour is described in Figure 2.2.*

We can interact with this machine and easily distinguish it from the above two machines by pushing the bad button. However, there is no trace that shows the difference from the above two machines.

This example shows that trace equivalence is too weak to be a good behavioural equivalence since it is even insensitive to deadlock. We should seek equivalences that are stronger than trace equivalence but weaker than graph isomorphism in terms of their distinguishing power.

2.2 Bisimulation

Let $\mathcal{R}$ be a binary relation, we use the infix notation $s \ \mathcal{R} \ s'$ to mean $(s, s') \in \mathcal{R}$, and we set $\mathcal{R}^{-1} = \{(s', s) \mid s \ \mathcal{R} \ s'\}$. The composition of relations $\mathcal{R}_1$ and $\mathcal{R}_2$ is $\mathcal{R}_1\mathcal{R}_2$, i.e. $s \ \mathcal{R}_1\mathcal{R}_2 \ s'$ holds if for some s'', both $s \ \mathcal{R}_1 \ s''$ and $s'' \ \mathcal{R}_2 \ s'$ hold.

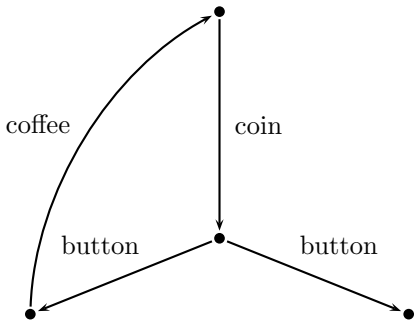


Figure 2.2: A bad vending machine

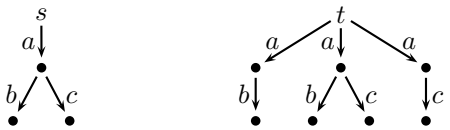


Figure 2.3: $s \not\sim t$

Definition 2.6 (Bisimulation) *A binary relation $\mathcal{R}$ on the states of an LTS is a* simulation *if whenever $s_1 \mathcal{R} s_2$:*

- *for all s_1' with $s_1 \xrightarrow{\alpha} s_1'$, there exists some s_2' such that $s_2 \xrightarrow{\alpha} s_2'$ and $s_1' \mathcal{R} s_2'$.*

The relation $\mathcal{R}$ is a bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.* Bisimilarity*, written $\sim$, is the union of all bisimulations; thus $s \sim t$ holds if there is a bisimulation $\mathcal{R}$ with $s \mathcal{R} t$. We write $s \prec t$ if there is a simulation $\mathcal{R}$ with $s \mathcal{R} t$.*

Example 2.7 *It is clear that $\prec$ is a preorder whose kernel $\prec \cap \prec^{-1}$ is coarser than bisimilarity. In general, the kernel does not coincide with bisimilarity. For example, consider the two states s and t in Figure 2.3. We have that $s \prec t$ and $t \prec s$, but $s \not\sim t$. However, the kernel coincides with bisimilarity for* deterministic *LTSs, where for any state s and action α there is at most one outgoing transition from s that is labelled by α.*

Bisimilarity enjoys a number of mathematical properties. Firstly, it is itself a bisimulation, and in fact is the largest bisimulation with respect to set inclusion. Secondly, it is an equivalence relation in that it is reflexive, symmetric and transitive. Finally, it can be characterised in terms of fixed point theory and a modal logic, as we shall see in the next few sections.

Proposition 2.8 *1. If $\mathcal{R}_1$ and $\mathcal{R}_2$ are bisimulations, then so are $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{R}_1\mathcal{R}_2$.*

2. $\sim$ is a bisimulation.

3. $\sim$ is an equivalence relation.

Definition 2.9 (Bisimulation up to $\sim$) *A binary relation $\mathcal{R}$ on the states of an LTS is a* bisimulation up to $\sim$ *if whenever $s_1 \mathcal{R} s_2$:*

- *for all s_1' with $s_1 \xrightarrow{\alpha} s_1'$, there exists some s_2' such that $s_2 \xrightarrow{\alpha} s_2'$ and $s_1' \sim \mathcal{R} \sim s_2'$;*
- *for all s_2' with $s_2 \xrightarrow{\alpha} s_2'$, there exists some s_1' such that $s_1 \xrightarrow{\alpha} s_1'$ and $s_1' \sim \mathcal{R} \sim s_2'$.*

Proposition 2.10 *If $\mathcal{R}$ is a bisimulation up to $\sim$ then $\mathcal{R} \subseteq \sim$.*

Proof: It is easy to check that $\sim \mathcal{R} \sim$ is a bisimulation and $\mathcal{R} \subseteq \sim \mathcal{R} \sim$. □

2.2.1 Bisimulation as post-fixed point

Let $(S, Act, \rightarrow)$ be an LTS. The function $F_\sim : \mathcal{P}(S \times S) \rightarrow \mathcal{P}(S \times S)$ is defined by letting $F_\sim(\mathcal{R})$ be the set of all pairs (s_1, s_2) such that

- for all s_1' with $s_1 \xrightarrow{\alpha} s_1'$, there exists some s_2' such that $s_2 \xrightarrow{\alpha} s_2'$ and $s_1' \ \mathcal{R} \ s_2'$;
- for all s_2' with $s_2 \xrightarrow{\alpha} s_2'$, there exists some s_1' such that $s_1 \xrightarrow{\alpha} s_1'$ and $s_1' \ \mathcal{R} \ s_2'$.

Proposition 2.11 *1. $F_\sim$ is monotone;*

2. $\mathcal{R}$ is a bisimulation iff $\mathcal{R}$ is a post-fixed point of $F_\sim$;

3. $\sim$ is the greatest fixed point of $F_\sim$.

2.2.2 Approximating bisimilarity

We can approximate bisimilarity using a family of inductively defined relations and their limit. Similarity can be approximated by similar constructions.

Definition 2.12 (Approximation of bisimilarity) *Let S be the state set of an LTS. We define:*

- $\sim_0 := S \times S$
- $s_1 \sim_{n+1} s_2$, *for $n \geq 0$, if*
  1. *for all s_1' with $s_1 \xrightarrow{\alpha} s_1'$, there exists some s_2' such that $s_2 \xrightarrow{\alpha} s_2'$ and $s_1' \sim_n s_2'$;*
  2. *for all s_2' with $s_2 \xrightarrow{\alpha} s_2'$, there exists some s_1' such that $s_1 \xrightarrow{\alpha} s_1'$ and $s_1' \sim_n s_2'$.*
- $\sim_\omega := \bigcap_{n \geq 0} \sim_n$

It is easy to see that $\sim$ is a finer relation than $\sim_\omega$. In fact it is strictly finer, i.e. $\sim \subsetneq \sim_\omega$, as the following example shows.

Example 2.13 *Consider the LTS $(S, \{a\}, \longrightarrow)$ where $S = \{s, t, \omega\} \cup \{0, 1, 2, ...\}$ and the transitions are:*

- *For all $n \geq 0$, $n+1 \xrightarrow{a} n$, $s \xrightarrow{a} n$, and $t \xrightarrow{a} n$;*
- *$\omega \xrightarrow{a} \omega$ and $t \xrightarrow{a} \omega$.*

It can be shown by induction on n that $s \sim_n t$ for all $n \geq 0$, thus $s \sim_\omega t$. However, $s \not\sim t$ because t can make a transition to reach the state ω which is not bisimilar to any state reachable from s.

In order to reach $\sim$ by approximation, in general we have to appeal to transfinite induction instead of the natural induction used in Definition 2.12. However, natural induction is sufficient if the LTSs are *image-finite*, that is, for all s the set of its derivatives $\{s' \mid s \xrightarrow{\alpha} s', \text{for some } \alpha\}$ is finite.

Proposition 2.14 *On image-finite LTSs, $\sim_\omega$ coincides with $\sim$.*

Proof: It is trivial to show by induction that $s \sim t$ implies $s \sim_n t$ for all $n \geq 0$, thus $s \sim_\omega t$.

Now we show that $\sim_\omega$ is a bisimulation. Suppose $s \sim_\omega t$ and $s \xrightarrow{a} s'$. We have to show that there is some t' with $t \xrightarrow{a} t'$ and $s' \sim_\omega t'$. Consider the set

$$T' := \{t' \mid t \xrightarrow{a} t' \wedge s' \not\sim_\omega t'\}.$$

For each $t' \in T'$, we have $s' \not\sim_\omega t'$, which means that there is some $n_{t'} > 0$ with $s' \not\sim_{n_{t'}} t'$. Since t is image-finite, T' is a finite set. Let $N = max\{n_{t'} \mid t' \in T'\}$. It holds that $s' \not\sim_N t'$ for all $t' \in T'$, since by a straightforward induction on m we can show that $s \sim_n t$ implies $s \sim_m t$ for all $m, n \geq 0$ with $n > m$. By the assumption $s \sim_\omega t$ we know that $s \sim_{N+1} t$. It follows that there is some t' with $t \xrightarrow{a} t'$ and $s' \sim_N t'$, so $t' \notin T'$ and hence $s' \sim_\omega t'$. By symmetry we also have that if $t \xrightarrow{a} t'$ then there is some s' with $s \xrightarrow{a} s'$ and $s' \sim_\omega t'$. □

2.2.3 Modal characterisation

Bisimulation provides a convenient proof technique (coinduction) to show that two processes are bisimilar since it suffices to construct a bisimulation that relates the two processes. However, to show them inequivalent, it is awkward to exhaustively prove that no possible bisimulation exists. For this purpose, we make use of modal logic to exhibit a modal formula satisfied by one process but not the other. Below we give a modal characterisation of strong bisimilarity in an infinitary Hennessy-Milner logic [HM85, vG01].

Definition 2.15 *The class $\mathcal{L}_B$ of* infinitary *Hennessy-Milner formulae over Act, ranged over by ϕ, is defined by the following grammar:*

$$\phi := \bigwedge_{i \in I} \phi_i \mid \langle a \rangle \phi \mid \neg \phi$$

where I is a set and $a \in Act$. We write $\top$ for $\bigwedge_{i \in \emptyset} \phi_i$. The satisfaction relation $\models \subseteq S \times \mathcal{L}_B$ *is defined by*

- $s \models \bigwedge_{i \in I} \phi_i$ *if* $s \models \phi_i$ *for all* $i \in I$.
- $s \models \langle a \rangle \phi$ *if for some* $s' \in S$, $s \xrightarrow{a} s'$ *and* $s' \models \phi$.
- $s \models \neg \phi$ *if it is not the case that* $s \models \phi$.

We write $s =_B t$ just when $s \models \phi \Leftrightarrow t \models \phi$ for all $\phi \in \mathcal{L}_B$.

Proposition 2.16 *$s \sim t$ iff $s =_B t$.*

Proof: ($\Rightarrow$) Suppose $s \sim t$, we show that $s \models \phi \Leftrightarrow t \models \phi$ by structural induction on ϕ.

- Let $s \models \langle a \rangle \phi$. Then $s \xrightarrow{a} s'$ and $s' \models \phi$ for some s'. Since $s \sim t$, there is some t' with $t \xrightarrow{a} t'$ and $s' \sim t'$. By induction hypothesis we have $t' \models \phi$, thus $t \models \langle a \rangle \phi$. By symmetry we also have $t \models \langle a \rangle \phi \Rightarrow s \models \langle a \rangle \phi$.
- Let $s \models \bigwedge_{i \in I} \phi_i$. Then $s \models \phi_i$ for all $i \in I$. So by induction $t \models \phi_i$, and we have $t \models \bigwedge_{i \in I} \phi_i$. By symmetry we also have $t \models \bigwedge_{i \in I} \phi_i$ implies $s \models \bigwedge_{i \in I} \phi_i$.
- Let $s \models \neg \phi$. So $s \not\models \phi$, and by induction we have $t \not\models \phi$. Thus $t \models \neg \phi$. By symmetry we also have $t \not\models \phi$ implies $s \not\models \phi$.

($\Leftarrow$) It sufficies to establish that $=_B$ is a bisimulation. Suppose $s =_B t$ and $s \xrightarrow{a} s'$. We have to show that there is some t' with $t \xrightarrow{a} t'$ and $s' =_B t'$. Consider the set

$$T' := \{t' \mid t \xrightarrow{a} t' \wedge s' \neq_B t'\}.$$

For each $t' \in T'$, we have $s' \neq_B t'$. It means that (i) either there is a formula $\phi_{t'}$ with $s' \models \phi_{t'}$ but $t' \not\models \phi_{t'}$ (ii) or there is a formula $\phi'_{t'}$ with $t' \models \phi'_{t'}$ but $s' \not\models \phi'_{t'}$. In the latter case we set $\phi_{t'} = \neg \phi'_{t'}$ and return back to the former case. Let

$$\phi := \langle a \rangle \bigwedge_{t' \in T'} \phi_{t'}.$$

It is clear that $s \models \phi$, hence $t \models \phi$ by $s =_B t$. It follows that there must be a t' with $t \xrightarrow{a} t'$ and $t' \notin T'$, which means $s' =_B t'$. □

In the above proof, the set T' is finite if we only consider image-finite LTSs. Then it sufficies to characterise similarity using modal formulae involving finite conjunctions. The sub-logic that only uses finite conjunctions is called Hennessy-Milner logic and it exactly captures similarity on image-finite LTSs.

2.2.4 Game characterisation

In this subsection we present a game characterisation of strong bisimilarity. The account below closely follows [Sti97].

Definition 2.17 (Bisimulation game) *Given an LTS $(S, Act, \rightarrow)$, a* strong bisimulation game $\mathcal{G}(s_0, t_0)$ *starting from the pair of states $(s_0, t_0) \in S \times S$ is a two-player game. A* play *of the game is a finite or infinite length sequence of the form $(s_0, t_0) \dots (s_i, t_i) \dots$. Player I, viewed as an attacker, attempts to show that the initial states are different whereas player II, viewed as a defender, wishes to establish that they are equivalent. Suppose an initial part of a play is $(s_0, t_0) \dots (s_i, t_i)$. The next pair (s_{i+1}, t_{i+1}) is determined by one the following two moves:*

- *Player I chooses a transition $s_i \xrightarrow{a} s_{i+1}$ and then player II chooses a transition with the same label $t_i \xrightarrow{a} t_{i+1}$.*
- *Player I chooses a transition $t_i \xrightarrow{a} t_{i+1}$ and then player II chooses a transition with the same label $s_i \xrightarrow{a} s_{i+1}$.*

The play continues with further moves. Player I always chooses first, and then player II, with full knowledge of player I's selection, must choose a corresponding transition from the other state.

A play of a game continues until one of the players wins. Player I wins if she can choose a transition and player II will be unable to match it. Player II wins if the play is infinite, or if the play reaches the configuration (s_i, t_i) and both states have no available transitions. In both these circumstances player I has been unable to detect a difference between the stating states.

Note that a game can have many different plays according to the choices made by the players. Player I can choose a state, an action and a transition. Player II can only choose one of the available transitions from the other state that is labelled with the same action as that chosen by player I. A play will be won either by player I or by player II; it cannot be won by both at the same time.

Example 2.18 *Consider the LTS given in Example 2.13. For the game $\mathcal{G}(t, \omega)$, there are plays that player I wins and plays that player II wins. If player I initially moves $t \xrightarrow{a} 0$ then after her opponent makes the move $\omega \xrightarrow{a} \omega$ we reach the configuration $(\omega, 0)$. Now player I wins by choosing the transition $\omega \xrightarrow{a} \omega$ because player II cannot answer a move from state 0. If player I initially chooses transitions $t \xrightarrow{a} \omega$ or $\omega \xrightarrow{a} \omega$ then player II can win. However player I has the power to win any play of (t, ω) by initially choosing the transition $t \xrightarrow{a} 0$.*

The above example shows that different plays of a game can have different winners. Nevertheless for each game one of the players is able to win any play irrespective of what moves her opponent makes. For explain this, we introduce the notion of strategy. A *strategy* for a player is a family of rules that tell the player how to move. For player I a rule has the form: if the play so far is $(s_0, t_0) \dots (s_i, t_i)$ then choose the transition ξ, where ξ is $s_i \xrightarrow{a} s'$ or $t_i \xrightarrow{a} t'$ for some states s', t'. For player II it has the form: if the play so far is $(s_0, t_0) \dots (s_i, t_i)$ and player I has chosen the transition ξ then choose the transition ξ', where ξ is $s_i \xrightarrow{a} s'$ or $t_i \xrightarrow{a} t'$ and ξ' is a corresponding transition of the other state. However it turns out that we only need to consider *history-free* strategies whose rules do not depend on what happened previously in the play. Therefore, for player I a rule has the form

$$\text{at configuration } (s, t) \text{ choose transition } \xi.$$

For player II a rule has the form

$$\text{at configuration } (s, t) \text{ when player I has chosen } \xi \text{ then choose } \xi'.$$

A player uses the strategy π in a play if all her moves obey the rules in π. The strategy π is a *winning strategy* if the player wins every play in which she uses π.

Example 2.19 *For the game $\mathcal{G}(t, \omega)$ we analysed in Example 2.18, player I has a winning strategy: if the configuration is (t, ω) then choose $t \xrightarrow{a} 0$, and if the configuration is $(0, \omega)$ then choose $\omega \xrightarrow{a} \omega$.*

Proposition 2.20 *For any game $\mathcal{G}(s,t)$ either player I or player II has a history-free winning strategy.*

Proof: See the proof of Proposition 3 in [Sti97]. □

If player II has a winning strategy for $\mathcal{G}(s,t)$ then s is *game equivalent* to t. Game equivalence is indeed an equivalence. Player II's winning strategy for $\mathcal{G}(s,s)$ is the copy-cat strategy (always choose the same transition as player I). If π is a winning strategy for player II for $\mathcal{G}(s,t)$ then π' which changes each rule "at configuration (s,t) choose ..." to "at configuration (t,s) choose ..." is a winning strategy for $\mathcal{G}(t,s)$. Finally if π is a winning strategy for player II for $\mathcal{G}(s,t)$ and π' for $\mathcal{G}(t,u)$ then the composition of these strategies is a winning strategy for player II for $\mathcal{G}(s,u)$: composition is defined in such a way that, for instance, the pair of rules "at (s,t) when player I has chosen $s \xrightarrow{a} s'$ choose ξ" in π and "at (t,u) when player I has chosen ξ choose ξ'" becomes " at (s,u) when player I has chosen $s \xrightarrow{a} s'$ choose ξ'".

When two states s and t are game equivalent, player II can always match player I's moves: if $s \xrightarrow{a} s'$ then there is a corresponding transition $t \xrightarrow{a} t'$ and s' and t' are also game equivalent, and if $t \xrightarrow{a} t'$ then there is also a corresponding transition $s \xrightarrow{a} s'$ with s' and t' game equivalent. This is precisely the criterion for being a bisimulation relation. The following proposition relates strong bisimilarity to the corresponding game characterisation.

Proposition 2.21 *s is game equivalent to t iff $s \sim t$.*

Proof: Assume that s is game equivalent to t. We show $s \sim t$ by establishing that the relation

$$\mathcal{R} = \{(s,t) \mid s \text{ and } t \text{ are game equivalent}\}$$

is a bisimulation. Suppose $s \xrightarrow{a} s'$, and as this is a possible move by player I we know that player II can respond with $t \xrightarrow{a} t'$ in such a way that $(s',t') \in \mathcal{R}$, and similarly when $t \xrightarrow{a} t'$. For the other direction suppose $s \sim t$, and so there is a bisimulation relation $\mathcal{R}$ such that $(s,t) \in \mathcal{R}$. We construct a winning strategy for player II for the game $\mathcal{G}(s,t)$: in any play, whatever move player I makes player II responds by making sure that the resulting pair of processes remain in the relation $\mathcal{R}$. Clearly player I cannot then win any play from $\mathcal{G}(s,t)$. □

The above game characterisation provides an intuitive understanding of bisimilarity. It can be used in practice to show that two states are strong bisimilar and to show that they are not.

2.3 Weak bisimulations

Sometimes we do not want to care about every action that a process performs, especially those internal or invisible actions. For this purpose we introduce a special action τ to represent invisible activities of processes and set $Act_\tau := Act \cup \{\tau\}$. We write $\Longrightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, i.e. $s \Longrightarrow s'$ if there are $s_0, ..., s_n \in S$ with $n \geq 0$ and $s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} ... \xrightarrow{\tau} s_n \equiv s'$. If in that sequence $n \geq 1$ then we write $s \overset{\tau}{\Longrightarrow} s'$. There are several versions of bisimulations, depending on how τ-transitions are ignored in the bisimulation game.

Definition 2.22 *A binary relation $\mathcal{R}$ on the states of an LTS is a* branching simulation *if $s \mathcal{R} t$ implies that whenever $s \xrightarrow{\alpha} s'$ then*

(C_τ): either $\alpha = \tau$ and $s' \mathcal{R} t$

(C_b): or there exist t', t'' such that $t \Longrightarrow t'' \xrightarrow{\alpha} t'$ with $s \mathcal{R} t''$ and $s' \mathcal{R} t'$.

The relation $\mathcal{R}$ is a branching bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are branching simulations. Two processes s and t are* branching bisimilar*, denoted $s \approx_b t$, if there exists a branching bisimulation relating s and t.*

There are some variants of the above matching conditions:

(C^s_τ): either $\alpha = \tau$ and there exists t' such that $t \Longrightarrow t'$ with $s \mathcal{R} t'$ and $s' \mathcal{R} t'$

(C^q_τ): either $\alpha = \tau$ and there exists t' such that $t \Longrightarrow t'$ with $s' \mathcal{R} t'$

(C_η): or there exist t', t'' such that $t \Longrightarrow t'' \stackrel{\alpha}{\longrightarrow\!\!\Longrightarrow} t'$ with $s \mathcal{R} t''$ and $s' \mathcal{R} t'$

(C_d): or there exists t' such that $t \Longrightarrow\!\stackrel{\alpha}{\longrightarrow} t'$ with $s' \mathcal{R} t'$

(C_w): or there exists t' such that $t \Longrightarrow\!\stackrel{\alpha}{\longrightarrow}\!\Longrightarrow t'$ with $s' \mathcal{R} t'$

Semi-branching bisimilarity *($\approx_s$) is defined in terms of (C^s_τ) and (C_b).*
quasi-branching bisimilarity *($\approx_q$) is defined in terms of (C^q_τ) and (C_b).*
η-bisimilarity *($\approx_\eta$) is defined in terms of (C_τ) and (C_η).*
Delay bisimilarity *($\approx_d$) is defined in terms of (C_τ) and (C_d).*
Weak bisimilarity *($\approx_w$) is defined in terms of (C_τ) and (C_w).*

It can be checked that all the bisimilarities defined above are indeed equivalence relations. In [vGW96] it is shown that $\approx_s$ coincides with $\approx_b$, the inclusions $\approx_b \subseteq \approx_q \subseteq \approx_\eta \subseteq \approx_w$ and $\approx_b \subseteq \approx_q \subseteq \approx_d \subseteq \approx_w$ are strict, but $\approx_\eta$ and $\approx_d$ are incomparable.

Although the largest semi-branching bisimulation coincides with the largest branching bisimulation, it is suggested in [Bas96] that semi-branching bisimulation is more intuitive than branching bisimulation. Moreover, to show that two processes are branching bisimilar, we often find it easier to exhibit a semi-branching bisimulation than a branching bisimulation. An example is the proof of Proposition 2.29.

In the above bisimulation game, if one process performs an initial τ-transition, the other can simulate it by doing nothing. If we disallow this by requiring the other process to make at least one step of τ-transitions, we obtain a slightly stronger relations.

Definition 2.23 *s and t are* quasi-branching congruent*, written $s \simeq_q t$, if*

1. *whenever $s \stackrel{\alpha}{\longrightarrow} s'$ then*
   (a) *either $\alpha = \tau$ and there exists t' such that $t \stackrel{\tau}{\Longrightarrow} t'$ and $s' \approx_q t'$,*
   (b) *or there exists t' such that $t \stackrel{\alpha}{\longrightarrow} t'$ and $s' \approx_q t'$;*
2. *symmetric to clause 1 by exchanging the roles of s and t.*

The other four congruences can be defined in a similar way.

Proposition 2.24 *For $x \in \{b, q, \eta, d, w\}$, if $s \approx_x t$ then one of the following three cases holds:*

1. *there exists some s' such that $s \stackrel{\tau}{\longrightarrow} s'$ and $s' \approx_x t$;*
2. *there exists some t' such that $t \stackrel{\tau}{\longrightarrow} t'$ and $s \approx_x t'$;*
3. *$s \simeq_x t$.*

Proof: Here we only consider $\approx_b$. Similar arguments can be made for other bisimilarities.

Let us examine how s and t match each other's transitions. We distinguish four possible cases.

1. $s \stackrel{\tau}{\longrightarrow} s'$ and $s' \approx_b t$. This is exactly what Clause 1 says.
2. $s \stackrel{\alpha}{\longrightarrow} s'$ and $t \stackrel{\tau}{\longrightarrow} t' \Longrightarrow t'' \stackrel{\alpha}{\longrightarrow} t'''$ such that $s \approx_b t''$ and $s' \approx_b t'''$. It follows from Lemma 2.28 that $s \approx_b t'$.
3. The symmetric cases of 1 and 2 by exchanging the roles of s and t.
4. None of the first three cases hold, i.e., either both s and t have no transition or each strong transition $s \stackrel{\alpha}{\longrightarrow} s'$ from s is matched by a strong transition $t \stackrel{\alpha}{\longrightarrow} t'$ from t with $s' \approx_b t'$ and vice versa. In this case it holds that $s \simeq t$.

□

2.4 "Bisimulation up to" techniques

To prove that two processes are bisimilar, usually one needs to exhibit a relation which is a bisimulation and which contains the two processes as a pair. In practice, to reduce the size of the relation exhibited one prefers to define a relation which is a bisimulation only when closed up under some specific relation, as to relieve the proof work needed. This kind of "bisimulation up to" techniques are first introduced in [Mil89b], for strong bisimulation and weak bisimulation. In a similar way, we present "bisimulation up to" techniques for branching bisimulation, quasi-branching bisimulation, η-bisimulation, as well as delay bisimulation.

Definition 2.25 *A branching bisimulation up to $\approx_b$ is a relation $\mathcal{R} \subseteq S \times S$ such that*

1. *if $s\mathcal{R}t$ and $s \Longrightarrow s' \xrightarrow{\alpha} s''$ then one of the following two cases holds:*
   (a) *$\alpha = \tau$ and there exists t'' such that $t \Longrightarrow t''$ with $s' \approx_b \mathcal{R} \approx_b t''$ and $s'' \approx_b \mathcal{R} \approx_b t''$;*
   (b) *there exist t', t'' such that $t \Longrightarrow t' \xrightarrow{\alpha} t''$ with $s' \approx_b \mathcal{R} \approx_b t'$ and $s'' \approx_b \mathcal{R} \approx_b t''$;*
2. *symmetric to Clause 1 by exchanging the roles of s and t.*

The other four "up to" relations can be defined similarly.

The following property is very simple, but it underpins several other results in the rest of the section.

Lemma 2.26 *For $x \in \{b, q, \eta, d, w\}$, if $s \approx_x t$. Then*

1. *if $s \Longrightarrow s'$ there exists some t' such that $t \Longrightarrow t'$ and $s' \approx_x t'$;*
2. *if $t \Longrightarrow t'$ there exists some s' such that $s \Longrightarrow s'$ and $s' \approx_x t'$.*

Proof: Clause 1 can be shown by induction on the length of the transition from s to s'; Clause 2 is similar.

□

In the literature on branching bisimulation, the *stuttering property* is widely used. A bisimulation satisfies this property means that: for any processes s_0 and s_k such that they are related to the same process t and s_0 can evolve into s_k by a sequence of τ-transitions, then all intermediate processes are also related to t. Following [Bas96], we give the formal definition.

Definition 2.27 (Stuttering property) *A binary relation $\mathcal{R}$ on S satisfies the* stuttering property *if for all $k \geq 0$ and $s, s_0, ..., s_k, t, t_0, ..., t_k \in S$:*

1. *$s_0\mathcal{R}t$, $s_k\mathcal{R}t$, and $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} ... \xrightarrow{\tau} s_k$ implies that $s_i\mathcal{R}t$ for all i, $1 \leq i < k$;*
2. *$s\mathcal{R}t_0$, $s\mathcal{R}t_k$, and $t_0 \xrightarrow{\tau} t_1 \xrightarrow{\tau} ... \xrightarrow{\tau} t_k$ implies that $s\mathcal{R}t_i$ for all i, $1 \leq i < k$.*

Lemma 2.28 *For $x \in \{b, q, \eta, d, w\}$, $\approx_x$ satisfies the stuttering property.*

Proof: It is shown in [vGW96] that $\approx_b$ satisfies the stuttering property. That proof can be easily adapted for $\approx_\eta$.

Below we only consider $\approx_w$; the cases for $\approx_q$ and $\approx_d$ are quite similar. Let $t \xrightarrow{\tau} t_1 \xrightarrow{\tau} ... \xrightarrow{\tau} t_n \xrightarrow{\tau} t'$ with $n \geq 1$ be a sequence of transitions from t satisfying $s \approx_w t$ and $s \approx_w t'$. We prove that

$$\mathcal{R} = \{(s, t_i) \mid 1 \leq i \leq n\} \cup \approx_w$$

is a weak bisimulation. It suffices to check that s and t_i can match each other's transition, for any $1 \leq i \leq n$.

1. If $s \xrightarrow{\alpha} s'$, we know from $s \approx_w t'$ that there are two possibilities:

(a) $\alpha = \tau$ and $s' \approx_w t'$. So $t_i \Longrightarrow \xrightarrow{\tau} t'$ and $s' \mathcal{R} t'$.

(b) $t' \Longrightarrow \xrightarrow{\alpha} \Longrightarrow t''$ and $s' \approx_w t''$. Hence $t_i \Longrightarrow t' \Longrightarrow \xrightarrow{\alpha} \Longrightarrow t''$ and $s' \mathcal{R} t''$.

2. If $t_i \xrightarrow{\alpha} t''$, then $t \Longrightarrow t_i \xrightarrow{\alpha} t''$. Since $t \approx_w s$ we know from Lemma 2.26 that there exists some s' such that $s \Longrightarrow s'$ and $t_i \approx_w s'$. Now there are two possibilities:

   (a) $\alpha = \tau$ and $t'' \approx_w s'$. Then either $s = s'$ with $t'' \mathcal{R} s$ or $s \Longrightarrow \xrightarrow{\tau} s'$ with $t'' \mathcal{R} s'$.

   (b) $s' \Longrightarrow \xrightarrow{\alpha} \Longrightarrow s''$ and $t'' \approx_w s''$. Therefore $s \Longrightarrow s' \Longrightarrow \xrightarrow{\tau} \Longrightarrow s''$ and $t'' \mathcal{R} s''$.

□

As expected, the "bisimulation up to" techniques are useful because of the following proposition.

Proposition 2.29 *For $x \in \{b, q, \eta, d, w\}$, if $\mathcal{R}$ is a x-bisimulation up to $\approx_x$ then $\mathcal{R} \subseteq \approx_x$.*

Proof: We first consider $\approx_b$. It suffices to prove that the relation $\approx_b \mathcal{R} \approx_b$ is a semi-branching bisimulation, from which it follows that $\mathcal{R} \subseteq \approx_b \mathcal{R} \approx_b \subseteq \approx_b$ by the coincidence between $\approx_s$ and $\approx_b$.

Suppose that $s_0 \approx_b s \mathcal{R} t \approx_b t_0$ and $s_0 \xrightarrow{\alpha} s_0''$. We distinguish two cases.

1. $\alpha = \tau$ and $s \Longrightarrow s''$ such that $s_0 \approx_b s''$ and $s_0'' \approx_b s''$. Let us analyze the transition $s \Longrightarrow s''$. There are two subcases.

   (a) $s = s''$. Then it is trivial to see that $t_0 \Longrightarrow t_0$, $s_0 \approx_b \mathcal{R} \approx_b t_0$ and $s_0'' \approx_b \mathcal{R} \approx_b t_0$.

   (b) There exists some s' such that $s \Longrightarrow s' \xrightarrow{\tau} s''$. By Lemma 2.28, we know that $s_0 \approx_b s'$. By Definition 2.25, we have two possible transitions:

      i. $t \Longrightarrow t''$ with $s' \approx_b \mathcal{R} \approx_b t''$ and $s'' \approx_b \mathcal{R} \approx_b t''$. By Lemma 2.26, there exists some t_0'' such that $t_0 \Longrightarrow t_0''$ and $t'' \approx_b t_0''$. It is clear that $s_0 \approx_b \mathcal{R} \approx_b t_0''$ and $s_0'' \approx_b \mathcal{R} \approx_b t_0''$.

      ii. $t \Longrightarrow t' \xrightarrow{\tau} t''$ with $s' \approx_b \mathcal{R} \approx_b t'$ and $s'' \approx_b \mathcal{R} \approx_b t''$. By Lemma 2.26, there exists some t_0' such that $t_0 \Longrightarrow t_0'$ and $t' \approx_b t_0'$. There are two possible transitions from t_0':

         A. $t_0' \Longrightarrow t_0''$ with $t' \approx_b t_0''$ and $t'' \approx_b t_0''$. It is easy to see that $t_0 \Longrightarrow t_0''$, $s_0 \approx_b \mathcal{R} \approx_b t_0''$ and $s_0'' \approx_b \mathcal{R} \approx_b t_0''$.

         B. $t_0' \Longrightarrow t_0''' \xrightarrow{\tau} t_0''$ with $t' \approx_b t_0'''$ and $t'' \approx_b t_0''$. It is easy to see that $t_0 \Longrightarrow t_0''' \xrightarrow{\tau} t_0''$, $s_0 \approx_b \mathcal{R} \approx_b t_0'''$ and $s_0'' \approx_b \mathcal{R} \approx_b t_0''$.

2. $s \Longrightarrow s' \xrightarrow{\alpha} s''$ with $s_0 \approx_b s'$ and $s_0'' \approx_b s''$. By Definition 2.25, we have two possible transitions:

   (a) $\alpha = \tau$ and $t \Longrightarrow t''$ with $s' \approx_b \mathcal{R} \approx_b t''$ and $s'' \approx_b \mathcal{R} \approx_b t''$. Then the arguments are the same as in Case 1(b)i.

   (b) $t \Longrightarrow t' \xrightarrow{\alpha} t''$ with $s' \approx_b \mathcal{R} \approx_b t'$ and $s'' \approx_b \mathcal{R} \approx_b t''$. The remaining arguments are similar to those in Case 1(b)ii.

The case for $\approx_\eta$ is similar to that of $\approx_b$.
For $\approx_q$ we can directly prove that $\approx_q \mathcal{R} \approx_q$ is a quasi-branching bisimulation.
The case for $\approx_d$ and $\approx_w$ are similar to that of $\approx_q$. □

2.5 A simple process algebra

Following [vGW96], we consider a simple language, the basic CCS. But the result in this section can be easily generalised. The class of processes $\mathcal{P}r$ is defined by the following syntax; we shall use $P, Q, ...$ as metavariables over $\mathcal{P}r$.

$P ::=$	$\mathbf{0}$	(inaction)
	$\alpha.P$	(prefixing, $\alpha \in A_\tau$)
	$P + Q$	(summation)

The process $\mathbf{0}$ represents a process that is unable to perform any action. $\alpha.P$ stands for a process that first performs the action α and then proceeds as P. $P+Q$ represents a process that behaves as either P or Q.

We define a labelled transition system $(\mathcal{P}r, Act, \longrightarrow)$, where the transition relation $\longrightarrow \subseteq \mathcal{P}r \times Act_\tau \times \mathcal{P}r$ is the smallest relation satisfying

- $\alpha.P \stackrel{\alpha}{\longrightarrow} P$;
- if $P \stackrel{\alpha}{\longrightarrow} R$ or $Q \stackrel{\alpha}{\longrightarrow} R$ then $P+Q \stackrel{\alpha}{\longrightarrow} R$.

The lemma below reports a simple property that holds for all the five bisimilarities defined in Section 2.3. We shall exploit this property to prove Lemma 2.35.

Lemma 2.30 *For $x \in \{b, q, \eta, d, w\}$, if $\tau.P + Q \approx_x P$ then $P + Q \approx_x P$.*

Proof: Here we only consider $\approx_b$. Similar arguments can be made for other bisimilarities.

We show that the relation

$$\mathcal{R} = \{(P+Q, P) \mid \tau.P + Q \approx_b P\} \cup Id$$

is a branching bisimulation up to $\approx_b$, where Id is the identity relation on $\mathcal{P}r$. For convenience of proof, we can regard $\approx_b$ as the largest semi-branching bisimulation. We only need to consider all transitions of $P+Q$ that are resulted from the transitions of Q.

1. Suppose $P+Q \stackrel{\alpha}{\longrightarrow} Q''$, which comes from $Q \stackrel{\alpha}{\longrightarrow} Q''$. Since $\tau.P+Q \approx_b P$, P must match the transition $\tau.P+Q \stackrel{\alpha}{\longrightarrow} Q''$. There are two cases.
   (a) $\alpha = \tau$ and $P \Longrightarrow P''$ with $\tau.P+Q \approx_b P''$ and $Q'' \approx_b P''$. It is easy to see that $P+Q \;\mathcal{R}\; P \approx_b \tau.P+Q \approx_b P''$. It follows that $P+Q \approx_b \mathcal{R} \approx_b P''$ and $Q'' \approx_b Id \approx_b P''$.
   (b) $P \Longrightarrow P' \stackrel{\alpha}{\longrightarrow} P''$ with $\tau.P+Q \approx_b P'$ and $Q'' \approx_b P''$. It is easy to see that $P+Q \approx_b \mathcal{R} \approx_b P'$ and $Q'' \approx_b Id \approx_b P''$.
2. Suppose $P+Q \stackrel{\tau}{\Longrightarrow} Q' \stackrel{\alpha}{\longrightarrow} Q''$, which comes from $Q \stackrel{\tau}{\Longrightarrow} Q' \stackrel{\alpha}{\longrightarrow} Q''$. Clearly we have the transition $\tau.P+Q \stackrel{\tau}{\Longrightarrow} Q'$. Since $\tau.P+Q \approx_b P$, by Lemma 2.26 there exists a transition $P \Longrightarrow P'$ such that $Q' \approx_b P'$. There are two ways for P' to match the transition $Q' \stackrel{\alpha}{\longrightarrow} Q''$.
   (a) $\alpha = \tau$ and $P' \Longrightarrow P''$ with $Q' \approx_b P''$ and $Q'' \approx_b P''$. Obviously we have that $P \Longrightarrow P''$ with $Q' \approx_b Id \approx_b P''$ and $Q'' \approx_b Id \approx_b P''$.
   (b) $P' \Longrightarrow P''' \stackrel{\alpha}{\longrightarrow} P''$ with $Q' \approx_b P'''$ and $Q'' \approx_b P''$. Obviously we can get $P \Longrightarrow P''' \stackrel{\alpha}{\longrightarrow} P''$ with $Q' \approx_b Id \approx_b P'''$ and $Q'' \approx_b Id \approx_b P''$.

□

For $\approx_d$ and $\approx_w$, we have the following result, where the part on $\approx_w$ is known as the original Hennessy Lemma in CCS.

Lemma 2.31 *For $x \in \{d, w\}$, $P \approx_x Q$ iff ($\tau.P \simeq_x Q$ or $P \simeq_x Q$ or $P \simeq_x \tau.Q$).*

Proof: For $\approx_w$, the result is proved in [Mil89b]. Further, that proof can be adapted for $\approx_d$ easily. □

Remark 2.32 *The above property does not hold for $\approx_b$, $\approx_q$ and $\approx_\eta$. For a counterexample, consider the two processes $\tau.(a+b)+a$ and $a+b$. Let $x \in \{b, q, \eta\}$, it is true that $\tau.(a+b)+a \approx_x a+b$. However,*

$$\tau.(\tau.(a+b)+a) \not\approx_x a+b \tag{2.1}$$

$$\tau.(a+b)+a \not\approx_x a+b \tag{2.2}$$

$$\tau.(a+b)+a \not\approx_x \tau.(a+b) \tag{2.3}$$

In (2.1) an action b from the right hind side cannot be matched up by any action from the left hide side of the inequality. Similar for (2.2). In (2.3) an action a from the left hind side cannot be matched up by any action from the right hide side of the inequality.

S1	$P + \mathbf{0}$	$=$	P
S2	$P + Q$	$=$	$Q + P$
S3	$P + (Q + R)$	$=$	$(P + Q) + R$
S4	$P + P$	$=$	P
B	$\alpha.(\tau.(P + Q) + Q)$	$=$	$\alpha.(P + Q)$
T1	$\alpha.\tau.P$	$=$	$\alpha.P$
T2	$\tau.P + P$	$=$	$\tau.P$
T3	$\alpha.(P + \tau.Q) + \alpha.Q$	$=$	$\alpha.(P + \tau.Q)$
T3′	$\tau.(P + \tau.Q) + \tau.Q$	$=$	$\tau.(P + \tau.Q)$

Table 2.1: All the axioms

2.6 Axiomatisations

An *(equational) axiomatisation* or an *axiom system* $\mathcal{A}$ for an equivalence $\simeq$ is a collection of equations $P = Q$. We write $\mathcal{A} \vdash P = Q$ if this equation can be derived from the equations in $\mathcal{A}$ by using the standard rules of equational logic. The *soundness* of $\mathcal{A}$ means that if $\mathcal{A} \vdash P = Q$ then $P \simeq Q$, and the *completeness* means that if $P \simeq Q$ then if $\mathcal{A} \vdash P = Q$, for all processes P and Q.

We shall consider complete axiomatisations of branching congruence, quasi-branching congruence, η-congruence, delay congruence, and weak congruence. For all the axiomatisations, the soundness properties are quite easy to show, thus we omit them. So we focus on the completeness properties and provide a uniform but simple completeness proof that works for the five congruences.

All the axioms that we need are displayed in Figure 2.1. It is shown in [Mil89b] that **S1-4** form a complete axiom system for strong bisimilarity. By adding the three τ-laws **T1-3**, Milner has obtained a complete axiom system for weak congruence. In [vGW96] van Glabbeek and Weijland have built a complete axiomatisation of branching congruence by adding **B** to **S1-4**. Two other congruences $\simeq_\eta$ and $\simeq_d$ are also axiomatized in [vGW96], just by adding {**B,T3**} and **T1-2**, respectively, to **S1-4**. The axiom **T3′** is the special case of **T3** when $\alpha = \tau$, and it is derivable from **T2** and **S4**. We shall show that $\simeq_q$ can be axiomatized by adding {**B,T3′**} to **S1-4**. We use the following abbreviations for the axiom systems of the five congruences.

$$\begin{array}{rcl} \mathcal{A}_b & = & \{\textbf{S1-4,B}\} \\ \mathcal{A}_q & = & \{\textbf{S1-4,B,T3}'\} \\ \mathcal{A}_\eta & = & \{\textbf{S1-4,B,T3}\} \\ \mathcal{A}_d & = & \{\textbf{S1-4,T1-2}\} \\ \mathcal{A}_w & = & \{\textbf{S1-4,T1-3}\} \end{array}$$

Note that **B** implies **T1** and is derivable from **T1-2**. So we can also use **T1** when doing equational reasoning in $\mathcal{A}_b, \mathcal{A}_q, \mathcal{A}_\eta$, and use **B** in $\mathcal{A}_d, \mathcal{A}_w$.

The following saturation properties, Clauses 1 and 3 in particular, are well-known in CCS. Here we also consider two special cases of the transition relation $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$: $\xrightarrow{\alpha}\Longrightarrow$ and $\Longrightarrow\xrightarrow{\alpha}$, in Clauses 2 and 3, respectively.

Lemma 2.33 (Saturation) *1. if* $P \stackrel{\tau}{\Longrightarrow} P'$ *then* $\{\textbf{S1-4,T3}'\} \vdash P = P + \tau.P'$*;*

2. if $P \xrightarrow{\alpha}\Longrightarrow P'$ *then* $\{\textbf{S1-4,T3}\} \vdash P = P + \alpha.P'$*;*

3. if $P \Longrightarrow\xrightarrow{\alpha} P'$ *then* $\{\textbf{S1-4,T2}\} \vdash P = P + \alpha.P'$*;*

4. if $P \Longrightarrow\xrightarrow{\alpha}\Longrightarrow P'$ *then* $\{\mathbf{S1\text{-}4,T2\text{-}3}\} \vdash P = P + \alpha.P'$.

Proof: The proof is carried out by transition induction. As an example, we show the second clause.
Base case: $P \xrightarrow{\alpha} P'$. Then the conclusion holds by **S4**.
Inductive step: $P \xrightarrow{\alpha}\Longrightarrow P'' \xrightarrow{\tau} P'$. Then we infer

$$\begin{array}{lll} \{\mathbf{S1\text{-}4,T3}\} \vdash P & = P + \alpha.P'' & \text{by induction} \\ & = P + \alpha.(P'' + \tau.P') & \text{by } \mathbf{S4} \\ & = P + \alpha.(P'' + \tau.P') + \alpha.P' & \text{by } \mathbf{T3} \\ & = P + \alpha.P' & \end{array}$$

□

As usual we use the notation of *normal form*. P is in normal form if it is of the form $\sum_{i=1}^{n} \alpha_i.P_i$, where each P_i is also in normal form.

Lemma 2.34 *For each* P*, there is a normal form* P' *such that* $\{\mathbf{S1\text{-}4}\} \vdash P = P'$.

We now introduce the important promotion lemma. It relates operational semantics to equational rewriting. Its proof is achieved by induction on the sizes of processes. We define the size, $size(P)$, of process P as follows.

$$\begin{array}{rcl} size(\mathbf{0}) & = & 0 \\ size(\alpha.P) & = & 1 + size(P) \\ size(P + Q) & = & size(P) + size(Q) \end{array}$$

Lemma 2.35 (Promotion) *1. If* $P \approx_b Q$ *then* $\mathcal{A}_b \vdash \tau.P = \tau.Q$;

2. If $P \approx_q Q$ *then* $\mathcal{A}_q \vdash \tau.P = \tau.Q$;

3. If $P \approx_\eta Q$ *then* $\mathcal{A}_\eta \vdash \tau.P = \tau.Q$;

4. If $P \approx_d Q$ *then* $\mathcal{A}_d \vdash \tau.P = \tau.Q$;

5. If $P \approx_w Q$ *then* $\mathcal{A}_w \vdash \tau.P = \tau.Q$.

Proof: By Lemma 2.34, we can assume that P and Q are in normal form.

1. We carry out the proof by induction on $size(P + Q)$.

   Base case If $size(P + Q) = 0$, then both P and Q are $\mathbf{0}$. We infer $\mathcal{A}_b \vdash \tau.\mathbf{0} = \tau.\mathbf{0}$ trivially.

   Inductive step Suppose $size(P+Q) > 0$. Since $P \approx_b Q$, by Proposition 2.24 we can distinguish three cases.

   (a) There exists some P' such that $P \xrightarrow{\tau} P'$ and $P' \approx_b Q$. To have the strong transition $P \xrightarrow{\tau} P'$, P must be of the form $\tau.P' + R$ for some process R. Since $\tau.P' + R \approx_b Q \approx_b P'$, it follows from Lemma 2.30 that $P' + R \approx_b P' \approx_b Q$. Note that $size(P' + R + Q) < size(\tau.P' + R + Q)$ and $size(P' + Q) < size(\tau.P' + R + Q)$. By induction hypothesis, it can be inferred that

   $$\mathcal{A}_b \vdash \tau.(P' + R) = \tau.Q = \tau.P'. \tag{2.4}$$

   So we derive

   $$\begin{array}{lll} \mathcal{A}_b \vdash \tau.P & = \tau.(\tau.P' + R) & \\ & = \tau.(\tau.(P' + R) + R) & \text{by (2.4)} \\ & = \tau.(P' + R) & \text{by } \mathbf{B} \\ & = \tau.Q & \text{by (2.4)} \end{array}$$

   (b) There exists some Q' such that $Q \xrightarrow{\tau} Q'$ and $P \approx_b Q'$. This case is symmetric to Case 1 by exchanging the roles of P and Q.

(c) $P \simeq Q$. We aim to prove that each summand of P can be absorbed by Q. Let $\alpha.P'$ be a summand of P, which gives rise to a transition $P \xrightarrow{\alpha} P'$. Correspondingly, there exists some Q' such that $\underline{Q \xrightarrow{\alpha} Q'}$ and $P' \approx_b Q'$. Clearly we can derive

$$\mathcal{A}_b \vdash Q = Q + \alpha.Q' \qquad (*)$$

by the axiom **S4**. Note that $size(P' + Q') < size(P + Q)$. So by induction hypothesis we obtain

$$\mathcal{A}_b \vdash \tau.P' = \tau.Q'. \qquad (2.5)$$

So we derive

$$\begin{array}{lll} \mathcal{A}_b \vdash Q + \alpha.P' & = Q + \alpha.\tau.P' & \text{by } \mathbf{T1} \\ & = Q + \alpha.\tau.Q' & \text{by (2.5)} \\ & = Q + \alpha.Q' & \text{by } \mathbf{T1} \\ & = Q & \text{by (*)} \end{array}$$

In summary, $\mathcal{A}_b \vdash Q + P = Q$ and symmetrically $\mathcal{A}_b \vdash P + Q = P$. Therefore $\mathcal{A}_b \vdash \tau.P = \tau.(P + Q) = \tau.Q$.

2. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_q$ and $\approx_q$, and replace the two underlined parts with "either $Q \xrightarrow{\alpha} Q'$ or $\alpha = \tau$ and $Q \stackrel{\tau}{\Longrightarrow} Q'$" and "either the axiom **S4** or Lemma 2.33(1)" respectively.

3. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_\eta$ and $\approx_\eta$, and replace the two underlined parts with $Q \xrightarrow{\alpha}\!\!\Longrightarrow Q'$ and Lemma 2.33(2) respectively.

4. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_d$ and $\approx_d$, and replace the two underlined parts with $Q \Longrightarrow\!\!\xrightarrow{\alpha} Q'$ and Lemma 2.33(3) respectively.

5. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_w$ and $\approx_w$, and replace the two underlined parts with $Q \Longrightarrow\!\!\xrightarrow{\alpha}\!\!\Longrightarrow Q'$ and Lemma 2.33(4) respectively.

□

Remark 2.36 *In the inductive step of the above proof, we have distinguished three independent cases by Proposition 2.24, and in the first two cases Lemma 2.30 plays an important role. Nevertheless, for behavioural equivalences that are insensitive to the branching structure of processes such as $\approx_d$ and $\approx_w$, the proof of the above lemma can be simplified. For instance, in the case of $\approx_w$, one just needs to consider two possibilities: (i) $P \xrightarrow{\alpha} P'$ with $P' \approx_w Q$, (ii) $P \xrightarrow{\tau} P'$ with $Q \Longrightarrow\!\!\xrightarrow{\alpha}\!\!\Longrightarrow Q'$ and $P' \approx_w Q'$. In the particular case of (ii), one can prove the property (*) by Lemma 2.33(4). This proof schema is adopted in [FY03] to show the promotion property of all the five weak behavioural equivalences considered in that paper. It is also adapted to a probabilistic setting in [DP05b] where probabilistic weak bisimilarity is investigated. However, the proof schema does not apply to weak behavioural equivalences that are to some extent sensitive to the branching structure of processes, such as $\approx_b$, $\approx_q$ and $\approx_\eta$. The reason is that for these equivalences the τ-transitions before the α-transition in (ii) cannot be simply absorbed. Otherwise, the branching structure of processes would not be observable.*

With the saturation and promotion properties we are now ready to establish the following completeness theorem.

Theorem 2.37 (Completeness) *1. If $P \simeq_b Q$ then $\mathcal{A}_b \vdash P = Q$;*

2. If $P \simeq_q Q$ then $\mathcal{A}_q \vdash P = Q$;

3. *If* $P \simeq_\eta Q$ *then* $\mathcal{A}_\eta \vdash P = Q$*;*

4. *If* $P \simeq_d Q$ *then* $\mathcal{A}_d \vdash P = Q$*;*

5. *If* $P \simeq_w Q$ *then* $\mathcal{A}_w \vdash P = Q$*.*

Proof:

1. Similar to the proof Lemma 2.35(1) we assume P, Q in normal form and proceed by induction on $size(P + Q)$. The base case is trivial, so we only consider the inductive step. Let $\alpha.P'$ be a summand of P. Then $P \xrightarrow{\alpha} P'$ must be matched up by $Q \xrightarrow{\alpha} Q'$ for some Q' such that $P' \approx_b Q'$. Clearly we can derive

$$\mathcal{A}_b \vdash Q = Q + \alpha.Q' \qquad (**)$$

by the axiom **S4**. By Promotion Lemma,

$$\mathcal{A}_b \vdash \tau.P' = \tau.Q'. \qquad (2.6)$$

Therefore

$$\begin{array}{lll} \mathcal{A}_b \vdash Q + \alpha.P' & = Q + \alpha.Q' & \text{by } \mathbf{T1} \text{ and } (2.6) \\ & = Q & \text{by } (**) \end{array}$$

Hence we have $\mathcal{A}_b \vdash Q + P = Q$. Symmetrically $\mathcal{A}_b \vdash P + Q = P$. Therefore $\mathcal{A}_b \vdash P = Q$.

2. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_q$ and $\approx_q$, and replace the two underlined parts with "either $Q \xrightarrow{\alpha} Q'$ or $\alpha = \tau$ and $Q \stackrel{\tau}{\Longrightarrow} Q'$" and "either the axiom **S4** or Lemma 2.33(1)" respectively.

3. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_\eta$ and $\approx_\eta$, and replace the two underlined parts with $Q \xrightarrow{\alpha}\!\!\Longrightarrow Q'$ and Lemma 2.33(2) respectively.

4. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_d$ and $\approx_d$, and replace the two underlined parts with $Q \Longrightarrow\!\!\xrightarrow{\alpha} Q'$ and Lemma 2.33(3) respectively.

5. The arguments are the same as in the proof of Clause 1 except that we change all the notations $\mathcal{A}_b$ and $\approx_b$ into $\mathcal{A}_w$ and $\approx_w$, and replace the two underlined parts with $Q \Longrightarrow\!\!\xrightarrow{\alpha}\!\!\Longrightarrow Q'$ and Lemma 2.33(4) respectively.

□

Remark 2.38 *For* $x \in \{d, w\}$*, there exists a even simpler completeness proof that does not rely on the Promotion Lemma. The reason is that Lemma 2.31 helps to lift* $P \approx_x Q$ *to either* $\tau.P \simeq_x Q$ *or* $P \simeq_x Q$ *or* $P \simeq_x \tau.Q$*, thus allowing the induction hypothesis to apply when proving (2.6) in the last proof. For instance, this is the method adopted in [Mil89b] for showing that* $\mathcal{A}_w$ *constitutes a complete axiom system for* $\simeq_w$*. For* $x \in \{b, q, \eta\}$*, however, this method cannot be used because the Hennessy Lemma fails for them (cf. Remark 2.32).*

2.7 Equivalence checking

2.7.1 A partition-refinement algorithm for bisimilarity

In this section we present an algorithm which, given an LTS $(S, A, \rightarrow)$ where both S and A are finite, iterately compute bisimilarity. It is due to Kanellakis and Smolka [KS90] and commonly known as a *partition refinement* algorithm (see Figure 2.4). The idea is to represent the state space as a set

$\mathcal{B} := \{S\}$
repeat
$\mathcal{B}_{old} := \mathcal{B}$
$\mathcal{B} :=$ **Refine**$(\mathcal{B})$
until $\mathcal{B}_{old} = \mathcal{B}$
return $\mathcal{B}$

Figure 2.4: Schema for the partition refinement algorithm

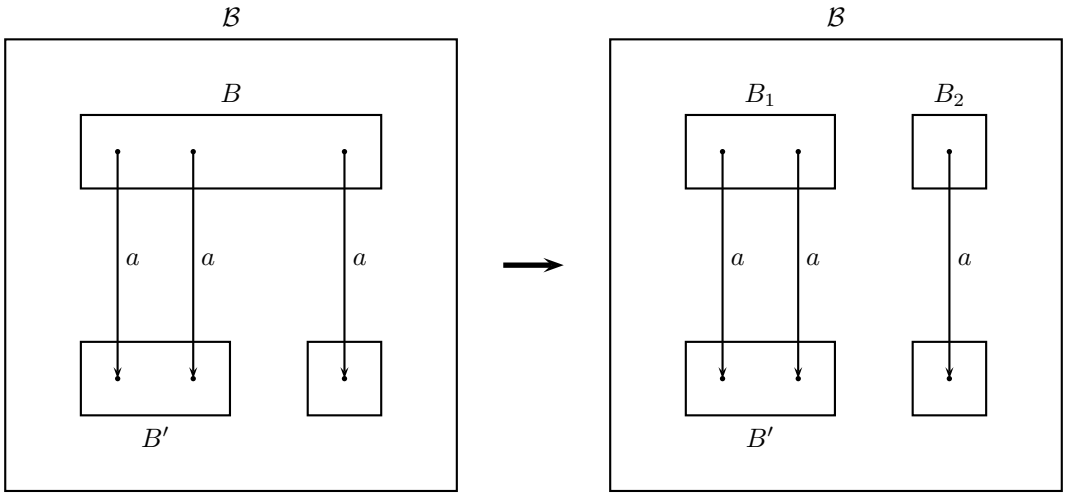


Figure 2.5: Splitting a block

of *blocks*, where a block is set of states standing for an equivalence class, and the equivalence of two given states can be tested by checking whether they belong to a same block. The blocks form a partition of the state space. Starting from the partition $\{S\}$, the algorithm iterately refines the partition by splitting each block into two smaller blocks if two states in one block are found to exhibit different behaviour. Eventually, when no further refinement is possible for a partition, the algorithm terminates and all states in a block of the partition are bisimilar.

Let $\mathcal{B} = \{B_1, ..., B_n\}$ be a partition consisting of a set of blocks. The algorithm tries to refine the partition by splitting each block. A *splitter* for a block $B \in \mathcal{B}$ is the block $B' \in \mathcal{B}$ such that some states in B have a-transitions, for $a \in A$, into B' and others do not. In this case B' splits B with respect to a into two blocks $B_1 = \{s \in B \mid \exists s' \in B' : s \xrightarrow{a} s'\}$ and $B_2 = B - B_1$.

The refinement operator **Refine**$(\mathcal{B})$ yields the partition

$$\textbf{Refine}(\mathcal{B}) = \bigcup_{B \in \mathcal{B}, a \in A} \textbf{Split}(B, a, \mathcal{B}) \tag{2.7}$$

where **Split**$(B, a, \mathcal{B})$ is the splitting procedure which detects whether the partition $\mathcal{B}$ contains a splitter for a given block $B \in \mathcal{B}$ with respect to action $a \in A$. If such splitter exists, B is splitted into two blocks B_1 and B_2. Otherwise, B itself is returned. In the procedure **Split** we use the following notation: $\{s \xrightarrow{a} \bullet\} := \{t \in S \mid s \xrightarrow{a} t\}$ is the *postset* of s respect to a, and $[S']_{\mathcal{B}} := \{B \in \mathcal{B} \mid \exists s \in S' : s \in B\}$ is the minimal set of blocks in $\mathcal{B}$ that cover all states in S'. So $[\{s \xrightarrow{a} \bullet\}]_{\mathcal{B}}$ is the set of blocks that are reachable from s by one step of a-transition. With a slight abuse of terminology we call this set the *postset of s in $\mathcal{B}$ with respect to a*. The procedure **Split** chooses a state from B and compares its postset in P with the postsets of other states in B. If the

postsets of two states are different, we know that there exists a splitter that will put these states in different blocks.

Procedure 1 Split$(B, a, \mathcal{B})$

```
choose s ∈ B
B1 = ∅
B2 = ∅
for all s' ∈ B do
  if [{s --a--> •}]_B = [{s' --a--> •}]_B then
    B1 = B1 ∪ {s'}
  else
    B2 = B2 ∪ {s'}
  end if
end for
if B2 = ∅ then
  return {B1}
else
  return {B1, B2}
end if
```

Procedure 2 Partition

```
B := {S}
changed := true
while changed do
  changed := false
  for all B ∈ B do
    for all a ∈ A do
      if Split(B, a, B) ≠ {B} then
        B := B − {B} ∪ Split(B, a, B)
        changed := true
        break
      end if
    end for
  end for
end while
```

In the main loop of **Partition**, the algorithm iteratively attempts to split every block in $\mathcal{B}$ with respect to every action in A until no further splitting is possible.

Given a partition $\mathcal{B}$, we write $\widehat{\mathcal{B}}$ for the equivalence relation generated by $\mathcal{B}$. For example, if $\mathcal{B} = \{\{s_1, s_2, s_3\}, \{s_4\}\}$, then

$$\widehat{\mathcal{B}} = \{(s_1, s_1), (s_2, s_2), (s_3, s_3), (s_1, s_2), (s_2, s_1), (s_1, s_3), (s_3, s_1), (s_2, s_3), (s_3, s_2), (s_4, s_4)\}.$$

Correctness of algorithm **Partition** relies on the fact that when *changed* is $false$, no block in $\mathcal{B}$ can be split. So $\widehat{\mathcal{B}} = F_{\sim}(\widehat{\mathcal{B}})$ and Proposition 2.11 (3) yields $\widehat{\mathcal{B}} \subseteq \sim$. Moreover, if we denote by $\widehat{\mathcal{B}}_n$ the partition after the n-th iteration of the main loop of **Partition**, we have $\sim \subseteq \sim_n \subseteq \widehat{\mathcal{B}}_n$. It follows that $\widehat{\mathcal{B}} = \sim$ when the algorithm terminates.

The complexity of **Partition** is given by the following theorem.

Theorem 2.39 *Given an LTS $(S, A, \rightarrow)$ with $|S| = n$ and $|\rightarrow| = m$, algorithm **Partition** takes $O(n \cdot m)$ time.*

Proof: The main loop of **Partition** is repeated at most n times. Within one iteration of the main loop, procedure **Split** is called for each block at most once for each action in A. In turn, **Split**

considers each transition of each state in the block at most once. So the calls to **Split** within one iteration of the main loop take $O(m)$ time. □

Performance of the partition refinement algorithm can be significantly improved by using more complex data structures. In addition, the algorithm can be used to decide several other behavioural equivalences such as weak bisimilarity, branching bisimilarity, testing equivalence etc. In order to obtain a partition refinement algorithm for an equivalence relation $\mathcal{R}$, one needs to define a suitable *transformation* $T_{\mathcal{R}}$ so that, for any two states s_1, s_2, $s_1 \mathcal{R} s_2$ if and only if $T_{\mathcal{R}}(s_1) \sim T_{\mathcal{R}}(s_2)$. See [CS01] for more details.

2.7.2 An on-the-fly algorithm for bisimilarity

The algorithm presented in the previous section requires to generate the whole state space of a system in advance. However, in many cases, one may be able to determine that one process fails to be related to another by examining only a fraction of the state space. We would like to have a verification algorithm that exploits this fact.

On-the-fly algorithms combine the verification of a system's behaviour with the generation of the system's state space. In this section we present the bisimilarity-checking algorithm originally proposed in [Lin98] for value-passing processes.

The main procedure in the algorithm is **Bisim**(s, t). It starts with the initial state pair (s, t), trying to find the smallest bisimulation relation containing the pair by matching transitions from each pair of states it reaches. It uses three auxiliary data structures:

- $NotBisim$ collects all state pairs that have already been detected as not bisimilar.
- $Visited$ collects all state pairs that have already been visited.
- $Assumed$ collects all state pairs that have already been visited and assumed to be bisimilar.

The core procedure, **Match**, is called from function **Bis** inside the main procedure **Bisim**. Whenever a new pair of states is encountered it is inserted into $Visited$. If two states fail to match each other's transitions then they are not bisimilar and the pair is added to $NotBisim$. If the current state pair has been visited before, we check whether it is in $NotBisim$. If this is the case, we return $false$. Otherwise, a loop has been detected and we make assumption that the two states are bisimilar, by inserting the pair into $Assumed$, and return $true$. Later on, if we find that the two states are not bisimilar after finishing searching the loop, then the assumption is wrong, so we first add the pair into $NotBisim$ and then raise the exception $WrongAssumption$, which forces the function **Bis** to run again, with the new information that the two states in this pair are not bisimilar. In this case, the size of $NotBisim$ has been increased by at least one. Hence, **Bis** can only be called for finitely many times. Therefore, the procedure **Bisim**(s, t) will terminate. If it returns $true$, then the set $(Visited - NotBisim)$ constitutes a bisimulation relation containing the pair (s, t).

Below we consider the time complexity of the algorithm. Let s and t be two states in an LTS with n states in total. The number of state pairs is bounded by n^2. The time required for the first call of **Bis**(s, t) is at most $O(n^2)$, for the second call is at most $O(n^2 - 1)$, $\cdots$. Therefore, the worst case time complexity of **Bis**(s, t) is $O(n^2 + (n^2 - 1) + \cdots + 1) = O(n^4)$. However, the worst case rarely happens in practical applications.

2.7.3 Tools

A few tools have been developed to check behavioural equivalences and preorders. Two noteworthy examples are the Concurrency Workbench [CPS93] and FDR [Ros94]. The former implements partition-refinement algorithms for several equivalences such as bisimulations, observational equivalence, and branching bisimulations. The latter is based on CSP [Hoa85a] and can be used to establish refinements (behavioural preorders) between processes.

Procedure 3 Bisim(s,t)

$NotBisim := \{\}$
fun Bis(s,t)={
  $Visited := \{\}$
  $Assumed := \{\}$
  Match(s,t)}
} **handle** $WrongAssumption \Rightarrow$ **Bis**(s,t)
return Bis(s,t)

Procedure 4 Match(s,t)

$Visited := Visisted \cup \{(s,t)\}$
$b = \bigwedge_{a \in A}$ **MatchAction**(s,t,a)
if $b = false$ **then**
  $NotBisim := NotBisim \cup \{(s,t)\}$
  if $(s,t) \in Assumed$ **then**
    raise $WrongAssumption$
  end if
end if
return b

Procedure 5 MatchAction(s,t,a)

for all $s \xrightarrow{a} s_i$ **do**
  for all $t \xrightarrow{a} t_j$ **do**
    $b_{ij} =$ **Close**(s_i, t_j)
  end for
end for
return $(\bigwedge_i(\bigvee_j b_{ij})) \wedge (\bigwedge_j(\bigvee_i b_{ij}))$

Procedure 6 Close(s,t)

if $(s,t) \in NotBisim$ **then**
  return $false$
else if $(s,t) \in Visited$ **then**
  $Assumed := Assumed \cup \{(s,t)\}$
  return $true$
else
  return Match(s,t)
end if

2.7.4 Formal verification

Checking behavioural equivalence between an implementation and a specification belongs to a broader research area called formal verification whose aim is to establish system correctness with mathematical rigour. In this section we give a briefly overview of some typical formal verification techniques. More details can be found in e.g. [BK08].

In developing complex software and hardware systems, more effort is spent on verification rather than on construction. Formal methods are playing more and more important roles in system verification. There are roughly two kinds of approaches in formal verification: deductive and model-based methods.

With *deductive* methods, the correctness of systems is formalised as properties in a mathematical theory, and then proven by tools such as theorem provers and proof checkers.

With *model-based* methods, real systems are modelled in a mathematical way (e.g. by automata) and then all states in system models are systematically explored. Typical examples include *model checking*, which explores state spaces exhaustively, and *simulation*, which works with a restrictive set of scenarios in systems models.

Model-based simulation Simulation is widely known and practically used in industry. The software tool called *simulator* allows the user to study how the system under consideration will react on certain scenarios. The scenarios are usually provided by the user or generated by tools like random scenario generators. Simulation is useful to assess the quality of prototype designs, but it is not good at finding subtle errors simply because of the impossibility of generating all possible scenarios of real systems.

Model checking Model checking explores the state space of a system model exhaustively (this is one of the reasons that it is mainly used in control-intensive applications and less suited for data-intensive applications since data often have infinite domains). The system model is usually expressed by finite state automata which can be generated from a model description specified in some appropriate dialect of programming languages like C or Java or hardware description languages such as Verilog or VHDL. The properties we are interested in can be described in a property specification language e.g. a temporal logic. Typical system properties are reachability (is a deadlock state reachable?), safety (something bad never happens), liveness (something good will eventually happen), fairness (will an event repeatedly occur under certain condition?), etc. With a system model and a desired property as input, a model checker is run to examine all relevant states to check if they satisfy the property. If a state is found to violate the property, the model checker provides a counterexample, which describes an execution path that goes from the initial state to a state that violates the property being verified.

When doing model checking, one often encounters the problem that a model is too large to be handled (state-space explosion). To combat this problem, some techniques have been developed that try to exploit implicit regularities in the structure of models; examples are the representation of state space using symbolic techniques such as binary decision diagrams or partial-order reduction. An alternative method is to use abstractions of system models; abstractions should be relatively small but preserve the (in)validity of the properties that are to be checked.

Nowadays, model checking is used in a wide range of applications such as embedded systems, software engineering, and hardware designs. Successful applications of model checking in system design have been reported by some companies such as IBM, Intel, Motorola, Lucent Technologies, Fujitsu etc.

Theorem proving Formal verification of some problems can also be regarded as proving theorems of the form: *system specification* $\Rightarrow$ *desired property*. This is the main idea with deductive methods that transform a system specification into a mathematical theory. A *theorem prover* uses a set of given axioms to either construct a proof of a theorem by generating the intermediate proof steps, or to refute it. Theorem provers are also called *proof assistants*. They have different variants.

Proof checkers are highly automated proof assistants. They check whether a proof suggested by a user is valid or not. The capacity of proof checkers to generate proofs automatically is very limited. *General-purpose proof assistants* usually have built-in search components. To reduce the search effort in theorem proving, interaction with the user is necessary. Interactive proof assistants are useful in giving a proof by keeping track of what needs to be done and by providing hints on how these remaining theorems can be proven. Moreover, each proof step is automatically verified.

The strength of theorem proving is that it can deal with infinite state space by using proof principles such as structural induction. The weakness is that the verification process is slow and error-prone. In addition, to use proof assistants requires users to have good expertise in mathematical logics. Typical proof assistants make use of higher-order logics; examples include PVS, Coq, HOL and Isabelle.

Chapter 3

A Brief Introduction of CCS and π-calculi

This chapter introduces some basic notions about process calculi. The presentation is based on CCS and the π-calculus, and partly guided by two textbooks [Mil99, SW01].

3.1 A calculus of communicating systems

We presuppose an infinite set of *process variables*, $Var = \{X, Y, ...\}$, and an infinite set of *names*, $\mathcal{N} = \{u, v, ...\}$. We use the set of *conames*, $\overline{\mathcal{N}} = \{\bar{u} \mid u \in \mathcal{N}\}$. Given a special name τ, we let a range over the set of *labels*, $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$. A label represents an indivisible action that a communicating system performs, such as reading a datum, or sending a datum. The class of *process expressions* $\mathcal{E}_{\mathsf{ccs}}$ is given by the following grammar:

$$E, F ::= \mathbf{0} \;\Big|\; a.E \;\Big|\; E + F \;\Big|\; E|F \;\Big|\; \nu u E \;\Big|\; X \;\Big|\; \mu_X E$$

The expression $\mathbf{0}$ represents *inaction*. The *prefix* $a.E$ describes the behaviour of first performing an action labelled a then behaving like E. The *sum* or *nondeterministic choice* $E + F$ behaves either like E or F nondeterministically. The *parallel composition* $E|F$ allows each of its components to behave independently, but also to synchronize with each other by a handshake on a complementary name. The *restriction* $\nu u E$ restricts the scope of u to E. The *recursion* $\mu_X E$ provides infinite behaviour by unfolding itself to be $E\{\mu_X E/X\}$. Operator precedence is (1) prefix, restriction, recursion, (2) parallel composition, and (3) nondeterministic choice.

Note that in CCS [Mil89b] the operators differ a little. The restriction $\nu u E$ is written $E\backslash u$. There is also a *renaming* operator $E[v_1/u_1, ..., v_n/u_n]$, which is not present here; its job is largely done by syntactic substitution of names. We shall write $E\{\widetilde{v}/\widetilde{u}\}$ for syntactic substitution of names $\widetilde{v}$ for names $\widetilde{u}$.

We use $fpv(E)$ for the set of free process variables (i.e., not bound by any μ_X) in E. As usual we identify expressions which differ only by a change of bound process variables. We shall write $E\{F_1, ..., F_n/X_1, ..., X_n\}$ or $E\{\widetilde{F}/\widetilde{X}\}$ for the result of simultaneously substituting F_i for each occurrence of X_i in E $(1 \leq i \leq n)$, renaming bound variables if necessary.

For operational semantics, we use a labelled transition system

$$(\mathcal{E}_{\mathsf{ccs}}, \mathcal{L}, \{\xrightarrow{a} \subseteq \mathcal{E}_{\mathsf{ccs}} \times \mathcal{E}_{\mathsf{ccs}} \;\Big|\; a \in \mathcal{L}\})$$

with $\mathcal{E}_{\mathsf{ccs}}$ as the set of states and $\mathcal{L}$ as transition labels. The transition relation is defined as the smallest relation generated by the rules in Table 3.1. The symmetric rules of sum1, par1 and com1 are omitted. As can be seen from the rule com1, for a communication between two processes to take place, one of them must offer an atomic action u, the other its complementary action $\bar{u}$. The

act	$\dfrac{}{a.E \xrightarrow{a} E}$	sum1	$\dfrac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'}$
par1	$\dfrac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F}$	com1	$\dfrac{E \xrightarrow{u} E' \quad F \xrightarrow{\bar{u}} F'}{E\|F \xrightarrow{\tau} E'\|F'}$
res	$\dfrac{E \xrightarrow{a} E'}{\nu u E \xrightarrow{a} \nu u E'}$ for $u \neq a$	rec	$\dfrac{E\{\mu_X E/X\} \xrightarrow{a} E'}{\mu_X E \xrightarrow{a} E'}$

Table 3.1: The transition rules for $\mathcal{E}_{\mathsf{ccs}}$

Figure 3.1: Link mobility

communication results in a τ-action, meaning that the communication serves as synchronisation and the result is invisible. On the other hand, in some literature on the analysis of distributed systems, parallel composition is defined as in CSP [Hoa85a], where a communication between two processes occurs if both of them offer the same action u, and the result is still a u-action.

3.2 The π-calculus

We first give the motivation and introduce the untyped π-calculus. Then we focus on channel types; we review sorts, simple channel types and subtyping progressively.

3.2.1 From CCS to the π-calculus

A significant limitation of CCS, as argued in [Mil99], is that it is not able to naturally specify communicating systems with dynamically changing connectivity. For example, let us consider the system composed of three components P, Q and R as displayed in Figure 3.1(1). Initially P and R are connected by the link a, while P and Q are connected by b. In the configuration of Figure 3.1(2), P and Q have evolved into P' and Q' respectively and the link to R has moved from P to Q. Since CCS gives us no way of creating new links among existing components, we are not able to specify the system in (1) as a CCS expression that can evolve into (2). However, this kind of evolution occurs often in many real systems. For instance, we may imagine R as a critical section that are accessed by P and Q successively. A natural way of dealing with link mobility like this is to give actions more structures so that links can be passed around in communicating systems. This is the method adopted by the π-calculus.

3.2.2 The untyped π-calculus

Let the set $\mathcal{N}$ of names be defined as in Section 3.1. The set $\mathcal{P}r_\pi$ of processes is defined by the following syntax:

$$P, Q ::= \mathbf{0} \mid u(x).P \mid \bar{u}v.P \mid P|Q \mid P + Q \mid \nu u P \mid !u(x).P$$

The *input prefix* $u(x).P$ can receive any name via u and continue as P with the received name substituted for x. The *output prefix* $\bar{u}v.P$ can send v via u and continue as P. The *replicated input* $!u(x).P$ can be thought of as an infinite composition $u(x).P|u(x).P|\cdots$, and it can encode recursive definitions [Mil91]. For example, take the simple CCS expression $E := \mu_X(u.(X|X))$, which has the infinite behaviour:

$$E \xrightarrow{u} E|E \xrightarrow{u} E|E|E \xrightarrow{u} ...$$

The same effect can be derived by using a replicated input:

$$\begin{array}{ll} & \nu v(\bar{v}|!v.u.(\bar{v}|\bar{v})) \\ \xrightarrow{\tau}\xrightarrow{u} & \nu v(\bar{v}|\bar{v}|!v.u.(\bar{v}|\bar{v})) \\ \xrightarrow{\tau}\xrightarrow{u} & \nu v(\bar{v}|\bar{v}|\bar{v}|!v.u.(\bar{v}|\bar{v})) \\ \xrightarrow{\tau}\xrightarrow{u} & ... \end{array}$$

All other operators (inaction, sum, restriction, and parallel composition) keep their meaning as in Section 3.1.

The π-calculus has two name-binding operators. In the processes $u(v).P$ and $\nu v P$ the occurrences of v in P are considered *bound* with scope P. An occurrence of a name in a process is *free* if it is not bound. We write $bn(P)$ (resp. $fn(P)$) for the set of names that have a bound (resp. free) occurrence in P. Changing a bound name into a fresh name is called *alpha-conversion*, and we identify processes up to alpha-conversion.

A substitution $\{v/u\}$ is a function on names that maps u to v and acts as identity on other names. Hence the postfix operator $P\{v/u\}$ is defined as the result of replacing all free occurrences of u in P by v, possibly applying alpha-conversion to avoid name capture by introducing unintended bound occurrences of names.

Convention: When considering a collection of processes and substitutions, we assume that each bound name of the processes is chosen to be unique, i.e., different from other names of the processes and the substitutions.

The early style [MPW92] of operational semantics for processes in $\mathcal{P}r_\pi$ is specified via a labelled transition system

$$(\mathcal{P}r_\pi, Act, \{\xrightarrow{\alpha} \subseteq \mathcal{P}r_\pi \times \mathcal{P}r_\pi \mid \alpha \in Act\})$$

where Act stands for the set of *actions*, of which there are four kinds.

1. The *internal action* τ. As in CCS, $P \xrightarrow{\tau} Q$ means that P can evolve into Q without any interaction with the environment. Internal actions arise from internal communication within a process.

2. An *input action* uv. The transition $P \xrightarrow{uv} Q$ means that P can receive v along u before evolving into Q. This departs from CCS because an input action contains the actual received value. Input actions arise from input prefixes.

3. A *free output action* $\bar{u}v$. The transition $P \xrightarrow{\bar{u}v} Q$ implies that P can emit the free name v along name u. Free output actions arise from output prefixes.

4. A *bound output action* $\bar{u}(v)$. Intuitively, $P \xrightarrow{\bar{u}(v)} Q$ means that P can emit the private name v (i.e. v is bound in P) along u before evolving into Q. Bound output actions arise from free output actions which carry names out of their scope, as in the process $\nu v(\bar{u}v.Q)$ for example.

kind	α	$subj(\alpha)$	$obj(\alpha)$	$fn(\alpha)$	$bn(\alpha)$
input	uv	u	v	$\{u,v\}$	$\emptyset$
free output	$\bar{u}v$	u	v	$\{u,v\}$	$\emptyset$
bound output	$\bar{u}(v)$	u	v	$\{u\}$	$\{v\}$
internal action	τ	-	-	$\emptyset$	$\emptyset$

Table 3.2: Terminology and notation for actions

$$\textsf{in}\ \frac{}{u(x).P \xrightarrow{uv} P\{v/x\}} \qquad \textsf{out}\ \frac{}{\bar{u}v.P \xrightarrow{\bar{u}v} P}$$

$$\textsf{sum1}\ \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \textsf{par1}\ \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\textsf{com1}\ \frac{P \xrightarrow{\bar{u}v} P' \quad Q \xrightarrow{uv} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \textsf{close1}\ \frac{P \xrightarrow{\bar{u}(v)} P' \quad Q \xrightarrow{uv} Q' \quad v \notin fn(Q)}{P|Q \xrightarrow{\tau} \nu v(P'|Q')}$$

$$\textsf{res}\ \frac{P \xrightarrow{\alpha} P' \quad u \notin n(\alpha)}{\nu u P \xrightarrow{\alpha} \nu u P'} \qquad \textsf{open}\ \frac{P \xrightarrow{\bar{u}v} P' \quad v \neq u}{\nu v P \xrightarrow{\bar{u}(v)} P'}$$

$$\textsf{rep}\ \frac{}{!u(x).P \xrightarrow{uv} !u(x).P|P\{v/x\}}$$

Table 3.3: The transition rules for $\mathcal{P}r_\pi$

Table 3.2 displays each kind of action, its *subject*, its *object*, its set of *free names*, and its set of *bound names*. We let $n(\alpha) := fn(\alpha) \cup bn(\alpha)$ denote the set of names occurring in α.

The transition relation $\xrightarrow{\alpha}$ is defined by the rules in Table 3.3. The symmetric rules of sum1, par1, com1 and close1 are omitted. Some of the rules deserve to be explained. We see from the rule in that $u(x).P$ can receive *any* name via u, and when a name is received it is substituted for the placeholder x in P. The rule open expresses extrusion of the scope of the name v, which can be seen in the rule close1. A process capable of performing a bound output $\bar{u}(v)$ can interact with a process that can receive v via u and in which v is not free. The interaction is represented by a τ-transition, and in the derivative the two components are within the scope of a restriction νv. We may say that the scope of v is opened via open while closed again via close1. The scope of the restricted name is extended to include the process that receives it. The side condition in the rule par1 is necessary because it prevents free names in Q from being incorrectly identified as bound names in P'. The rule rep captures the idea that $!u(x).P$ can spawn infinitely many copies of $u(x).P$ and each copy can perform an input action as in the rule in.

Sometimes we use the notation $\stackrel{\alpha}{\Longrightarrow}$ which is an abbreviation for $(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^*$, where $(\xrightarrow{\tau})^*$ is the reflexive and transitive closure of $\xrightarrow{\tau}$.

The capacity to change the connectivity of a network of processes is the crucial difference between the π-calculus and CCS. Let us consider an example based on Figure 3.1. Suppose two processes P, Q need to use some resource R in a critical section. Initially only process P has access to the resource, represented by a communication link a. After an interaction with Q along other link b this access is transferred to Q. This kind of behaviour can be described in the π-calculus as follows: process P that sends a along b is $\bar{b}a.P'$ (suppose a does not appear in P'); process Q that receives some link along b and then uses it to send data c is $b(x).\bar{x}c.Q''$. The interaction between P and Q is formulated as:

$$\bar{b}a.P'|b(x).\bar{x}c.Q'' \xrightarrow{\tau} P'|\bar{a}c.Q''.$$

After the interaction, the connection between P and R disappears while a new connection between Q' and R is built, where Q' is the process $\bar{a}c.Q''$.

The π-calculus presented above is *monadic* in that a message consists of exactly one name. Sometime we want to send messages consisting of more than one name. So it is useful to allow *polyadic* inputs and outputs: $u(x_1, ..., x_n).P$ and $\bar{u}\langle v_1, ..., v_n\rangle.Q$. Accordingly we can extend the transition rules in Table 3.3 to allow for polyadic communication:

$$u(\widetilde{x}).P|\bar{u}\langle\widetilde{v}\rangle.Q \xrightarrow{\tau} P\{\widetilde{v}/\widetilde{x}\}|Q$$

where $\widetilde{x}$ and $\widetilde{v}$ have the same length. After the extension we obtain the polyadic π-calculus [Mil91].

3.2.3 Sorts and sorting

To regulate the use of names, Milner introduced the notion *sorting* [Mil91], which is essential to avoid disagreement in the arities of tuples carried by a given name in the polyadic π-calculus. Assume a basic collection Σ of *sorts*. To every name u is assigned a sort ι, and we write $u : \iota$. A *sort list* over Σ is a finite sequence $\widetilde{\iota} = \iota_1, ..., \iota_n$ of sorts. Σ^* is the set of all sort lists over Σ. We write $\widetilde{u} : \widetilde{\iota}$ if $u_i : \iota_i$ for all i with $1 \leq i \leq n$. A *sorting* over Σ is a partial function

$$f : \Sigma \mapsto \Sigma^*$$

and we say that a process respects f if, for every subterm of the form $u(\widetilde{v}).P$ or $\bar{u}\langle\widetilde{v}\rangle.Q$,

$$\text{if } u : \iota \text{ then } \widetilde{v} : f(\iota).$$

For example, consider the following process

$$F \;:=\; !a(n, b).\texttt{if}\ \ n = 1\ \texttt{then}\ \bar{b}\langle 1\rangle\ \texttt{else}\ \nu c(\bar{a}\langle n - 1, c\rangle | c(m).\bar{b}\langle m * n\rangle) \tag{3.1}$$

which represents the factorial function. Let us choose $\Sigma = \{S_a, S_{bc}, \texttt{Nat}\}$ with

$$a : S_a,\ b : S_{bc},\ c : S_{bc},\ m : \texttt{Nat},\ n : \texttt{Nat}.$$

Then a sorting f respected by F is such that

$$f : \begin{cases} S_a & \mapsto & \texttt{Nat}, S_{bc} \\ S_{bc} & \mapsto & \texttt{Nat}. \end{cases}$$

3.2.4 A simple example

Before proceeding to the formal presentation of type systems for the π-calculus, we informally explain the usefulness of types, capability types in particular, by a simple example from [PS96]. Imagine the common situation in which two processes must cooperate in the use of a shared resource such as a printer. The printer provides a request channel u on which the client processes send their data for printing. If one client process has the form $Q_1 := \bar{u}v_1.\bar{u}v_2.\mathbf{0}$, then we expect that executing the program $\nu u(P|Q_1|Q_2)$ should result in the print jobs represented by v_1 and v_2 eventually being received and processed, in that order, by the printer process P (see Figure 3.2(1), where an arrow from one process to another means that some data are transmitted from the source of the arrow to the target). However, this is not necessarily the case: a misbehaving implementation of Q_2 can disrupt the protocol expected by P and Q_1 simply by reading print requests from u and throwing them away: $Q_2 :=!u(v).\mathbf{0}$ (see Figure 3.2(2)). We can prevent this kind of bad behaviour by distinguishing three kinds of access to a channel – the capability to write values, the capability to read values, and the capability to do both – and requiring each process to use its channels with some prescribed capabilities. Here, for instance, the client processes should only be allowed to write to u. The printer, on the other hand, should only read from u. When we impose this constraint, process Q_2 will be ruled out because it attempts to read from u.

Figure 3.2: A printer example

T	::=	$V \mid L$	types
V	::=	$L \mid \texttt{bool} \mid \texttt{Nat}$	value types
L	::=	$\#V$	channel types
Γ	::=	$\emptyset \mid \Gamma, x : T$	type environments
w	::=	$x \mid true, false \mid 0, 1, 2, \cdots$	values
P, Q	::=	$\mathbf{0} \mid u(x : V).P \mid \bar{u}w.P \mid P\|Q \mid P + Q \mid (\nu a : L)P \mid !u(x : V).P$	processes

$$\textsf{T-in}\ \frac{\Gamma \vdash u : \#V \quad \Gamma, x : V \vdash P}{\Gamma \vdash u(x : V).P} \qquad \textsf{T-out}\ \frac{\Gamma \vdash u : \#V \quad \Gamma \vdash w : V \quad \Gamma \vdash P}{\Gamma \vdash \bar{u}w.P} \qquad \textsf{T-nil}\ \frac{}{\Gamma \vdash \mathbf{0}}$$

$$\textsf{T-par}\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P|Q} \qquad \textsf{T-sum}\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P + Q} \qquad \textsf{T-res}\ \frac{\Gamma, a : L \vdash P}{\Gamma \vdash (\nu a : L)P}$$

$$\textsf{T-rep}\ \frac{\Gamma \vdash u(x : V).P}{\Gamma \vdash !u(x : V).P}$$

Table 3.4: Processes, types and typing rules of the simply typed π-calculus

3.2.5 The simply typed π-calculus

To begin with, we introduce some terminology and notation concerning types. An *assignment* of a type T to a name u is of the form $u : T$. A *type environment* is a finite set of assignments of types to names, where the names in the assignments are all different. We use Γ, Δ to range over type environments. Sometimes we regard a type environment Γ as a partial function from names to types. Thus we write $\Gamma(u)$ for the type assigned to u by Γ, and say that the names of the assignments in Γ are the names on which Γ is defined. We write $dom(\Gamma)$ for the set of names of the assignments in Γ. When $dom(\Gamma) \cap dom(\Delta) = \emptyset$, we write Γ, Δ for the union of Γ and Δ.

A process type judgment $\Gamma \vdash P$ asserts that process P is well typed under the type environment Γ, and a value type judgment $\Gamma \vdash w : V$ that value w has type V under the type assumptions in Γ. We say P is well typed under Γ if the judgment $\Gamma \vdash P$ can be derived by using the typing rules of a given type system.

A *channel* is a name that may be used to engage in communications. The *values* are the objects that can be exchanged along channels. The *channel types* are the types that can be ascribed to channels. The *value types* are the types that can be ascribed to values. In the π-calculus, channel types can be used as value types. In other words, we allow channels to be transmitted as values, and hence allow mobility.

Since our purpose in this section is to introduce the type system of the simply typed π-calculus rather than to propose a pragmatic notation for programming, we adopt an explicitly typed presentation in which every bound name is annotated with a type. The syntax of types and processes as

well as the typing rules are shown in Table 3.4. The syntactic distinction between value types and channel types is made by the use of V to range over value types and L over channel types (the letter C is reserved for other use later). However, in typing and operational rules, unless important for the sense we will use only the letters S, T, which stand for arbitrary types. We observe that in the simply typed π-calculus there is only one channel type constructor $\#V$. A type assignment $u : \#V$ means that u can be used as a channel to carry values of type V. Value types include channel types and *basic types*, thus both channels and basic values are allowed to be communicated. In the above table, we only display the typing rules for processes. The typing rules for values are the usual ones. For example, we may have the following rules:

$$\frac{}{\Gamma, x : T \vdash x : T} \qquad \frac{}{\Gamma \vdash \mathit{true} : \mathtt{bool}} \qquad \frac{}{\Gamma \vdash 0 : \mathtt{Nat}} \qquad \ldots$$

For simplicity we only consider two basic types: `bool`, for boolean values, and `Nat`, for natural numbers. Values of basic types are said to be of first-order because, unlike channels, they cannot carry other values. We also assume some basic operations on first-order values. For example, we may use addition $(n+m)$, subtraction $(n-m)$, multiplication $(n*m)$ for `Nat` expressions. To avoid being too specific, we do not give a rigid syntax and typing rules for first-order expressions. We just assume a separate mechanism for evaluating expressions of type `Nat`.

The inert process $\mathbf{0}$ is well typed under any type environment. The parallel composition and the sum of two processes are well typed if each is well typed in isolation. A process $(\nu a : L)P$ is well typed if P observes the constraints imposed both by the type environment and by the declared type L of the new name a. Note that here L is a channel type. In an input $u(x : V).P$ the subject u should have a channel type, which is compatible with the type of x, moreover, the body P is well typed under the extension of Γ with the type of x. The case for $!u(x : V).P$ is similar. An output $\bar{u}w.P$ is well typed if u has a channel type compatible with that of w, and P itself is well typed.

The transition rules for typed processes are similar to those of the untyped processes (Table 3.3). We just need to annotate bound names with their types. For example, the rule `open` would take this form:

$$\frac{P \xrightarrow{(\nu \widetilde{v}:\widetilde{V})\bar{u}w} P' \quad a \in \mathit{fn}(w) \backslash \{\widetilde{v}, u\}}{(\nu a : L)P \xrightarrow{(\nu \widetilde{v}:\widetilde{V}, a:L)\bar{u}w} P'}$$

Given the operational semantics for typed processes, we can prove the *subject reduction* property, which represents the fact that type judgments are invariant under computation. In particular, if $\Gamma \vdash P$ and $P \xrightarrow{\tau} P'$ then it holds that $\Gamma \vdash P'$.

3.2.6 Subtyping

Subtyping is a preorder on types. If S is a subtype of T then all operations available on values of type T are also available on values of type S; therefore an expression of type S can always replace an expression of type T. The possibility of having operations that work on all subtypes of a given type is a major advantage of subtyping in a programming language.

We shall write *subtype judgments* in the form $S <: T$, which asserts that S is a subtype of T (equally T is a supertype of S). A type construct is *covariant* in its i-th argument if the construct preserves the direction of subtyping in that argument. Dually, a type construct is *contravariant* in its i-th argument if the construct inverts the direction of subtyping in that argument. A type construct is *invariant* in its i-th argument if it is both covariant and contravariant in that argument.

We now refine channel types by distinguishing between the capabilities of using a channel in input or in outputs. For this we introduce the types $\mathtt{i}V$ and $\mathtt{o}V$, with the intended meanings: $\mathtt{i}V$ is the type of a channel that can be used only in input and that carries values of type V; similar for $\mathtt{o}V$ w.r.t. output. By extending the simply typed π-calculus with the two capability types, we obtain the *simply typed π-calculus with subtyping*. For this, we redefine channel types as

$$L ::= \#V \mid \mathtt{i}V \mid \mathtt{o}V \qquad \text{channel types}$$

$$\textsf{S-ref}\ \frac{}{T <: T} \qquad \textsf{S-tra}\ \frac{T <: T' \quad T' <: T''}{T <: T''} \qquad \textsf{S-bi}\ \frac{}{\#T <: \mathtt{i}T}$$

$$\textsf{S-bo}\ \frac{}{\#T <: \mathtt{o}T} \qquad \textsf{S-ii}\ \frac{T <: T'}{\mathtt{i}T <: \mathtt{i}T'} \qquad \textsf{S-oo}\ \frac{T <: T'}{\mathtt{o}T' <: \mathtt{o}T}$$

$$\textsf{S-bb}\ \frac{T <: T' \quad T' <: T}{\#T <: \#T'}$$

$$\textsf{T-ins}\ \frac{\Gamma \vdash u : \mathtt{i}V \quad \Gamma, x : V \vdash P}{\Gamma \vdash u(x : V).P} \qquad \textsf{T-outs}\ \frac{\Gamma \vdash u : \mathtt{o}V \quad \Gamma \vdash w : V \quad \Gamma \vdash P}{\Gamma \vdash \bar{u}w.P}$$

$$\textsf{subsum}\ \frac{\Gamma \vdash u : T \quad T <: T'}{\Gamma \vdash u : T'}$$

(rules T-ins and T-outs replace T-in and T-out respectively)

Table 3.5: Additional rules on subtyping

and use the additional rules reported in Table 3.5.

We briefly comment on the subtyping rules. The rules S-ref and S-tra show that $<:$ is a preorder. The axioms S-bi and S-bo show that a name of all capabilities can be used in places where only the input or only the output capability is required. Rule S-ii says that `i` is a covariant construct, while S-oo says that `o` is a contravariant construct. Finally S-bb shows that # is invariant.

The typing rules T-ins and T-outs are similar to the rules T-in and T-out, except that now the subject of a prefix is checked to have the appropriate input or output capability. The old rules are derivable from the new ones.

Chapter 4

Probabilistic Process Algebras

4.1 Probabilistic LTS

A (discrete) probability distribution over a set S is a mapping $\Delta : S \to [0,1]$ with $\sum_{s\in S} \Delta(s) = 1$. The *support* of Δ is given by $\lceil \Delta \rceil := \{\, s \in S \mid \Delta(s) > 0 \,\}$. In this chapter we only consider finite state systems, so it suffices to use distributions with finite support; let $\mathcal{D}(S)$, ranged over by Δ, Θ, Φ, denote the collection of all such distributions over S. We use $\overline{s}$ to denote the point distribution, satisfying

$$\overline{s}(t) = \begin{cases} 1 & \text{if } t = s, \\ 0 & \text{otherwise} \end{cases}$$

while if $p_i \geq 0$ and Δ_i is a distribution for each i in some finite index set I, then $\sum_{i\in I} p_i \cdot \Delta_i$ is given by

$$(\sum_{i\in I} p_i \cdot \Delta_i)(s) \quad = \quad \sum_{i\in I} p_i \cdot \Delta_i(s)$$

If $\sum_{i\in I} p_i = 1$ then this is easily seen to be a distribution in $\mathcal{D}(S)$, and we will sometimes write it as $p_1 \cdot \Delta_1 + \ldots + p_n \cdot \Delta_n$ when the index set I is $\{1, \ldots, n\}$, and as $\Delta_1 \,{}_p\!\oplus\, \Delta_2$ when $I = \{p_1, p_2\}$ and $p = p_1$. Finally, the *product* of two probability distributions Δ, Θ over S, T is the distribution $\Delta \times \Theta$ over $S \times T$ defined by $(\Delta \times \Theta)(s,t) := \Delta(s) \cdot \Theta(t)$.

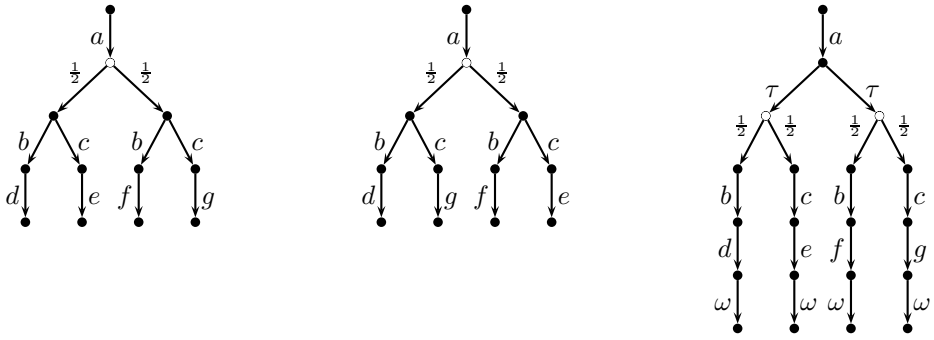
For Δ a distribution over S and function $f : S \to X$ into a vector space X; we sometimes write $\mathrm{Exp}_\Delta(f)$ for $\sum_{s\in S} \Delta(s) \cdot f(s)$, the *expected value* of f. Our primary use of this notation is with X being the vector space of reals, but we will also use it with tuples of reals, or distributions over some set. In the latter case this amounts to the notation $\sum_{i\in I} p_i \cdot \Delta_i$, where I is a finite index set and $\sum_{i\in I} p_i = 1$. When $p \in [0,1]$, we also write $f_1 \,{}_p\!\oplus\, f_2$ for $p \cdot f_1 + (1-p) \cdot f_2$. More generally, for function $F : S \to \mathcal{P}^+(X)$ with $\mathcal{P}^+(X)$ being the collection of non-empty subsets of X, we define $\mathrm{Exp}_\Delta F := \{\, \mathrm{Exp}_\Delta(f) \mid f \in F \,\}$; here $f \in F$ means that $f : S \to X$ is a *choice function*, that is it satisfies the constraint that $f(s) \in F(s)$ for all $s \in S$.

We can now present the probabilistic generalisation of an LTS:

Definition 4.1 *A* probabilistic labelled transition system *(pLTS) is a triple* $\langle S, Act_\tau, \rightarrow \rangle$*, where*

1. *S is a set of states and Act is a set of actions, as in LTSs;*
2. *Act_τ is a set of external actions Act with an internal action τ;*
3. *$\rightarrow \subseteq S \times Act_\tau \times \mathcal{D}(S)$.*

As with LTSs, we usually write $s \xrightarrow{\alpha} \Delta$ in place of $(s, \alpha, \Delta) \in \rightarrow$. We write $s \xrightarrow{\alpha}$ for $\exists \Delta : s \xrightarrow{\alpha} \Delta$ and $s \rightarrow$ for $\exists \alpha : s \xrightarrow{\alpha}$. An LTS may be viewed as a degenerate pLTS, one in which only point distributions are used.



(*i*) P (*ii*) Q (*iii*) T

Figure 4.1: Example pLTSs

Given that in a pLTS transitions go from states to distributions, we need to introduce additional edges to connect distributions back to states, thereby obtaining a bipartite graph. States are represented by nodes of the form $\bullet$ and distributions by nodes of the form $\circ$. For any state s and distribution Δ with $s \xrightarrow{\alpha} \Delta$ we draw an edge from s to Δ, labelled with α. Consequently, the edges leaving a $\bullet$-node are all labelled with actions from Act_τ. For any distribution Δ and state s in $\lceil\Delta\rceil$, the support of Δ, we draw an edge from Δ to s, labelled with $\Delta(s)$. Consequently, the edges leaving a $\circ$-node are labelled with positive real numbers that sum up to 1.

Some example pLTSs are described in Figure 4.1. Note that to make these graphs more compact we omit nodes $\circ$ when they represent trivial point distributions.

For each state s, the outgoing transitions $s \xrightarrow{\alpha} \Delta$ represent the nondeterministic alternatives in the state s. The nondeterministic choices provided by s are supposed to be resolved by the environment, which are formalized by a *scheduler* or an *adversary*. On the other hand, the probabilistic choices in the underlying distribution Δ are made by the system itself. Therefore, for each state s, the environment chooses some outgoing transition $s \xrightarrow{\alpha} \Delta$. Then the action α is performed, the system resolves the probabilistic choice, and subsequently with probability $\Delta(s')$ the system reaches state s'.

If we impose the constraint that for any state s and action α at most one outgoing transition from s is labelled α, then we obtain *reactive* pLTSs which are probabilistic counterpart to *deterministic* LTS. The assumption behind reactive pLTSs is that the environment determines which actions are possible. For each action α, the system behaves randomly and reaches state s' with probability $\Delta(s')$. Formally, a pLTS is reactive if for each $s \in S, \alpha \in Act$ we have that $s \xrightarrow{\alpha} \Delta$ and $s \xrightarrow{\alpha'} \Delta'$ imply $\Delta = \Delta'$.

4.2 Probabilistic bisimulations

4.2.1 Lifting relations

Definition 4.2 *Given two sets S and T and a relation $\mathcal{R} \subseteq S \times T$. We lift $\mathcal{R}$ to be a relation $\overline{\mathcal{R}} \subseteq \mathcal{D}(S) \times \mathcal{D}(T)$ by letting $\Delta \,\overline{\mathcal{R}}\, \Theta$ whenever*

1. *$\Delta = \sum_{i \in I} p_i \cdot \overline{s_i}$, where I is a finite index set and $\sum_{i \in I} p_i = 1$*
2. *For each $i \in I$ there is a state t_i such that $s_i \mathcal{R} t_i$*
3. *$\Theta = \sum_{i \in I} p_i \cdot \overline{t_i}$.*

An important point here is that in the decomposition (1) of Δ into $\sum_{i \in I} p_i \cdot \overline{s_i}$, the states s_i are *not necessarily distinct*: that is, the decomposition is not in general unique. Thus when establishing

the relationship between Δ and Θ, a given state s in Δ may play a number of different roles. This is reflected in the following property.

Proposition 4.3 $\overline{s}\ \overline{\mathcal{R}}\ \Theta$ *iff* $s\ \mathcal{R}\ t$ *for all* $t \in \lceil\Theta\rceil$. □

From Definition 4.2, the next two properties follows. In fact, they are sometimes used in the literature as definitions of lifting relations instead of being properties (see e.g. [SL94, LS91]).

Proposition 4.4 *1. Let* Δ *and* Θ *be distributions over* S *and* T*, respectively. Then* $\Delta\ \overline{\mathcal{R}}\ \Theta$ *iff there exists a weight function* $w : S \times T \to [0,1]$ *such that*

(a) $\forall s \in S : \sum_{t\in T} w(s,t) = \Delta(s)$

(b) $\forall t \in T : \sum_{s\in S} w(s,t) = \Theta(t)$

(c) $\forall (s,t) \in S \times T : w(s,t) > 0 \Rightarrow s\ \mathcal{R}\ t$.

2. Let Δ, Θ *be distributions over a finite set* S *and* $\mathcal{R}$ *is an equivalence relation. Then* $\Delta\ \overline{\mathcal{R}}\ \Theta$ *iff* $\Delta(C) = \Theta(C)$ *for all equivalence class* $C \in S/\mathcal{R}$*, where* $\Delta(C)$ *stands for the accumulation probability* $\sum_{s\in C} \Delta(s)$.

□

Proposition 4.5 *1. If* $\mathcal{R}_1 \subseteq \mathcal{R}_2$ *then* $\overline{\mathcal{R}_1} \subseteq \overline{\mathcal{R}_2}$

2. If $\mathcal{R}$ *is a transitive relation, then so is* $\overline{\mathcal{R}}$.

Proof:

1. By definition, it is straightforward to show that if $\Delta_1\ \overline{\mathcal{R}_1}\ \Delta_2$ and $\mathcal{R}_1 \subseteq \mathcal{R}_2$ then $\Delta_1\ \overline{\mathcal{R}_2}\ \Delta_2$.

2. Given three distributions $\Delta_1, \Delta_2, \Delta_3$ and a transitive relation $\mathcal{R}$, we show that if $\Delta_1\ \overline{\mathcal{R}}\ \Delta_2$ and $\Delta_2\ \overline{\mathcal{R}}\ \Delta_3$ then $\Delta_1\ \overline{\mathcal{R}}\ \Delta_3$.

   First $\Delta_1\ \overline{\mathcal{R}}\ \Delta_2$ means that

$$\Delta_1 = \sum_{i\in I} p_i \cdot \overline{s_i}, \qquad s_i\ \mathcal{R}\ s'_i, \qquad \Delta_2 = \sum_{i\in I} p_i \cdot \overline{s'_i}\ ; \tag{4.1}$$

   also $\Delta_2\ \overline{\mathcal{R}}\ \Delta_3$ means that

$$\Delta_2 = \sum_{j\in J} q_j \cdot \overline{t'_j}, \qquad t'_j\ \mathcal{R}\ t_j, \qquad \Delta_3 = \sum_{j\in J} q_j \cdot \overline{t_i}\ ; \tag{4.2}$$

and we can assume *w.l.o.g.* that all the coefficients p_i, q_j are non-zero. Now define $I_j = \{\, i \in I \mid s'_i = t'_j \,\}$ and $J_i = \{\, j \in J \mid t'_j = s'_i \,\}$, so that trivially

$$\{(i,j) \mid i \in I,\ j \in J_i\} \qquad = \qquad \{(i,j) \mid j \in J,\ i \in I_j\} \tag{4.3}$$

and note that

$$\Delta_2(s'_i) = \sum_{j\in J_i} q_j \qquad \text{and} \qquad \Delta_2(t'_j) = \sum_{i\in I_j} p_i \tag{4.4}$$

Because of (4.4) we have

$$\begin{array}{ll} \Delta_1 = \sum_{i\in I} p_i \cdot \overline{s_i} & = \sum_{i\in I} p_i \cdot \sum_{j\in J_i} \frac{q_j}{\Delta_2(s'_i)} \cdot \overline{s_i} \\ & = \sum_{i\in I} \sum_{j\in J_i} \frac{p_i \cdot q_j}{\Delta_2(s'_i)} \cdot \overline{s_i} \end{array}$$

Similarly

$$\begin{array}{ll} \Delta_3 = \sum_{j\in J} q_j \cdot \overline{t_j} & = \sum_{j\in J} q_j \cdot \sum_{i\in I_j} \frac{p_i}{\Delta_2(t'_j)} \cdot \overline{t_j} \\ & = \sum_{j\in J} \sum_{i\in I_j} \frac{p_i \cdot q_j}{\Delta_2(t'_j)} \cdot \overline{t_j} \\ & = \sum_{i\in I} \sum_{j\in J_i} \frac{p_i \cdot q_j}{\Delta_2(t'_j)} \cdot \overline{t_j} \qquad \text{by (4.3)} \end{array}$$

Now for each j in J_i we know that in fact $t'_j = s'_i$, and so from the middle parts of (4.1) and (4.2), together with the transitivity of $\mathcal{R}$, we obtain $\Delta_1 \; \overline{\mathcal{R}} \; \Delta_3$. □

In analogy with Definition 4.2, the transition relation $\xrightarrow{a}$ between states and distributions can be lifted to one between distributions and distributions, by letting $\Delta \xrightarrow{\hat{a}} \Theta$ whenever

1. $\Delta = \sum_{i \in I} p_i \cdot \overline{s_i}$, where I is a finite index set and $\sum_{i \in I} p_i = 1$
2. For each $i \in I$ there is a distribution Θ_i such that $s_i \xrightarrow{a} \Theta_i$
3. $\Theta = \sum_{i \in I} p_i \cdot \Theta_i$.

The lifting construction satisfies the following two useful properties, whose proofs we leave to the reader.

Proposition 4.6 Suppose $\mathcal{R} \subseteq S \times S$ or $S \times \mathcal{D}(S)$ and $\sum_{i \in I} p_i = 1$. Then we have

1. $\Theta_i \; \overline{\mathcal{R}} \; \Delta_i$ implies $(\sum_{i \in I} p_i \cdot \Theta_i) \; \overline{\mathcal{R}} \; (\sum_{i \in I} p_i \cdot \Delta_i)$.
2. If $(\sum_{i \in I} p_i \cdot \Theta_i) \; \overline{\mathcal{R}} \; \Delta$ then $\Delta = \sum_{i \in I} p_i \cdot \Delta_i$ for some set of distributions Δ_i such that $\Theta_i \; \overline{\mathcal{R}} \; \Delta_i$.

□

4.2.2 Probabilistic bisimulation

Definition 4.7 *A relation $\mathcal{R} \subseteq S \times S$ is a* strong simulation *if $s \; \mathcal{R} \; t$ implies*

- *if $s \xrightarrow{a} \Delta$ then there exists some Θ such that $\overline{t} \xrightarrow{a} \Theta$ and $\Delta \; \overline{\mathcal{R}} \; \Theta$.*

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong simulations, then $\mathcal{R}$ is a strong bisimulation. *We $s \sim_p t$ if there exists a strong bisimulation $\mathcal{R}$ with $s \; \mathcal{R} \; t$.*

Lemma 4.8 Let $\mathcal{R}$ be a bisimulation. Suppose $\Delta \; \overline{\mathcal{R}} \; \Phi$ and $\Delta \xrightarrow{a} \Delta'$. Then $\Phi \xrightarrow{a} \Phi'$ for some Φ' such that $\Delta' \; \overline{\mathcal{R}} \; \Phi'$.

Proof: First $\Delta \; \overline{\mathcal{R}} \; \Phi$ means that

$$\Delta = \sum_{i \in I} p_i \cdot \overline{s_i}, \qquad s_i \; \mathcal{R} \; r_i, \qquad \Phi = \sum_{i \in I} p_i \cdot \overline{r_i} \; ; \tag{4.5}$$

also $\Delta \xrightarrow{a} \Delta'$ means

$$\Delta = \sum_{j \in J} q_j \cdot \overline{t_j}, \qquad t_j \xrightarrow{a} \Theta_j, \qquad \Delta' = \sum_{j \in J} q_j \cdot \Theta_j \; , \tag{4.6}$$

and we can assume *w.l.o.g.* that all the coefficients p_i, q_j are non-zero. Now define $I_j = \{ \, i \in I \mid s_i = t_j \, \}$ and $J_i = \{ \, j \in J \mid t_j = s_i \, \}$, so that trivially

$$\{(i,j) \mid i \in I, \; j \in J_i\} \qquad = \qquad \{(i,j) \mid j \in J, \; i \in I_j\} \tag{4.7}$$

and note that

$$\Delta(s_i) = \sum_{j \in J_i} q_j \qquad \text{and} \qquad \Delta(t_j) = \sum_{i \in I_j} p_i \tag{4.8}$$

Because of (4.8) we have

$$\begin{array}{rl} \Phi = \sum_{i \in I} p_i \cdot \overline{r_i} & = \sum_{i \in I} p_i \cdot \sum_{j \in J_i} \frac{q_j}{\Delta(s_i)} \cdot \overline{r_i} \\ & = \sum_{i \in I} \sum_{j \in J_i} \frac{p_i \cdot q_j}{\Delta(s_i)} \cdot \overline{r_i} \end{array}$$

Now for each j in J_i we know that in fact $t_j = s_i$, and so from the middle parts of (4.5) and (4.6) we obtain $\overline{r_i} \xrightarrow{a} \Phi'_{ij}$ such that $\Theta_j \; \overline{\mathcal{R}} \; \Phi'_{ij}$. So we have that

$$\Phi \xrightarrow{a} \Phi' \quad = \sum_{i \in I} \sum_{j \in J_i} \frac{p_i \cdot q_j}{\Delta(s_i)} \cdot \Phi'_{ij}$$

where within the summations $s_i = t_j$, so that, using (4.7), Φ' can also be written as

$$\sum_{j\in J}\sum_{i\in I_j}\frac{p_i\cdot q_j}{\Delta(t_j)}\cdot\Phi'_{ij} \tag{4.9}$$

Below we show that $\Delta' \,\overline{\mathcal{R}}\, \Phi'$, which we do by manipulating Δ' so that it takes on a form similar to that in (4.9):

$$\begin{array}{ll}\Delta' & = \sum_{j\in J} q_j\cdot\Theta_j \\ & = \sum_{j\in J} q_j\cdot\sum_{i\in I_j}\frac{p_i}{\Delta(t_j)}\cdot\Theta_j \qquad \text{using (4.8) again} \\ & = \sum_{j\in J}\sum_{i\in I_j}\frac{p_i\cdot q_j}{\Delta(t_j)}\cdot\Theta_j\end{array}$$

Comparing this with (4.9) above we see that the required result, $\Delta' \,\overline{\mathcal{R}}\, \Phi'$, follows from an application of Proposition 4.6(1). □

4.2.3 Modal characterisation

Definition 4.9 *The class $\mathcal{L}_{PB}$ of modal formulae over Act, ranged over by ϕ, is defined by the following grammar:*

$$\phi := \bigwedge_{i\in I}\phi_i \mid \langle a\rangle\phi \mid \neg\phi \mid \bigoplus_{i\in I} p_i\cdot\phi_i$$

The satisfaction relation $\models\subseteq \mathcal{D}(S)\times\mathcal{L}_{PB}$ *is defined by*

- $\Delta\models\bigwedge_{i\in I}\phi_i$ *if* $\Delta\models\phi_i$ *for all* $i\in I$.
- $\Delta\models\langle a\rangle\phi$ *if for some* $\Delta'\in\mathcal{D}(S)$, $\Delta\xrightarrow{a}\Delta'$ *and* $\Delta'\models\phi$.
- $\Delta\models\neg\phi$ *if it is not the case that* $\Delta\models\phi$.
- $\Delta\models\bigoplus_{i\in I} p_i\cdot\phi_i$ *if there are* $\Delta_i\in\mathcal{D}(S)$, *for all* $i\in I, t\in\lceil\Delta_i\rceil$, *with* $\overline{t}\models\phi_i$, *such that* $\Delta=\sum_{i\in I} p_i\cdot\Delta_i$.

We write $\Delta =_{PB}\Theta$ just when $\Delta\models\phi\Leftrightarrow\Theta\models\phi$ for all $\phi\in\mathcal{L}_{PB}$.

Proposition 4.10 *$s\sim_p t$ iff $\overline{s}=_{PB}\overline{t}$.*

Proof: ($\Rightarrow$) We prove a general result that

$$\text{If } \Delta\,\overline{\sim_p}\,\Theta \text{ then } \Delta =_{PB}\Theta \tag{4.10}$$

from which it is immediately that $s\sim_p t$ implies $\overline{s}=_{PB}\overline{t}$.

Suppose $\Delta\,\overline{\sim_p}\,\Theta$, we show that $\Delta\models\phi\Leftrightarrow\Theta\models\phi$ by structural induction on ϕ.

- Let $\Delta\models\langle a\rangle\phi$. Then $\Delta\xrightarrow{a}\Delta'$ and $\Delta'\models\phi$ for some Δ'. Since $\Delta\,\overline{\sim_p}\,\Theta$, there is some Θ' with $\Theta\xrightarrow{a}\Theta'$ and $\Delta'\,\overline{\sim_p}\,\Theta'$. By induction hypothesis we have $\Theta'\models\phi$, thus $\Theta\models\langle a\rangle\phi$. By symmetry we also have $\Theta\models\langle a\rangle\phi\Rightarrow\Delta\models\langle a\rangle\phi$.
- Let $\Delta\models\bigwedge_{i\in I}\phi_i$. Then $\Delta\models\phi_i$ for all $i\in I$. So by induction $\Theta\models\phi_i$, and we have $\Theta\models\bigwedge_{i\in I}\phi_i$. By symmetry we also have $\Theta\models\bigwedge_{i\in I}\phi_i$ implies $\Delta\models\bigwedge_{i\in I}\phi_i$.
- Let $\Delta\models\neg\phi$. So $\Delta\not\models\phi$, and by induction we have $\Theta\not\models\phi$. Thus $\Theta\models\neg\phi$. By symmetry we also have $\Theta\not\models\phi$ implies $\Delta\not\models\phi$.
- Let $\Delta\models\bigoplus_{i\in I} p_i\cdot\phi_i$. So $\Delta=\sum_{i\in i} p_i\cdot\Delta_i$ and for all $i\in I$ and $t\in\lceil\Delta_i\rceil$ we have $\overline{t}\models\phi_i$. Since $\Delta\,\overline{\sim_p}\,\Theta$, by Proposition 4.6 (2) we have that $\Theta=\sum_{i\in I} p_i\cdot\Theta_i$ and $\Delta_i\,\overline{\sim_p}\,\Theta_i$. It follows that for each $t'\in\lceil\Theta_i\rceil$ there is some $t\in\lceil\Delta_i\rceil$ with $t\sim_p t'$, thus $\overline{t}\,\overline{\sim_p}\,\overline{t'}$. So by induction we have $\overline{t'}\models\phi_i$ for all $t'\in\lceil\Theta_i\rceil$ and $i\in I$. Therefore, we have $\Theta\models\bigoplus_{i\in I} p_i\cdot\phi_i$.

($\Leftarrow$) Let $\mathcal{R} = \{(s,t) \mid \overline{s} =_{PB} \overline{t}\}$. We show that $\mathcal{R}$ is a strong probabilistic bisimulation. Suppose $s \mathcal{R} t$ and $s \xrightarrow{a} \Delta$. We have to show that there is some Θ with $\overline{t} \xrightarrow{a} \Theta$ and $\Delta \overline{\sim_p} \Theta$. Consider the set

$$T := \{\Theta \mid \overline{t} \xrightarrow{a} \Theta \wedge \Theta = \sum_{s' \in \lceil \Delta \rceil} \Delta(s') \cdot \Theta_{s'} \wedge \exists s' \in \lceil \Delta \rceil, \exists t' \in \lceil \Theta_{s'} \rceil : \overline{s'} \neq_{PB} \overline{t'}\}$$

For each $\Theta \in T$, there must be some $s'_\Theta \in \lceil \Delta \rceil$ and $t'_\Theta \in \lceil \Theta_{s'_\Theta} \rceil$ such that (i) either there is a formula ϕ_Θ with $\overline{s'_\Theta} \models \phi_\Theta$ but $\overline{t'_\Theta} \not\models \phi_\Theta$ (ii) or there is a formula ϕ'_Θ with $\overline{t'_\Theta} \models \phi'_\Theta$ but $\overline{s'_\Theta} \not\models \phi'_\Theta$. In the latter case we set $\phi_\Theta = \neg\phi'_\Theta$ and return back to the former case. So for each $s' \in \lceil \Delta \rceil$ it holds that $\overline{s'} \models \bigwedge_{\{\Theta \in T \mid s'_\Theta = s'\}} \phi_\Theta$ and for each $\Theta \in T$ with $s'_\Theta = s'$ there is some $t'_\Theta \in \lceil \Theta_{s'} \rceil$ with $\overline{t'_\Theta} \not\models \bigwedge_{\{\Theta \in T \mid s'_\Theta = s'\}} \phi_\Theta$. Let

$$\phi := \langle a \rangle \bigoplus_{s' \in \lceil \Delta \rceil} \Delta(s') \cdot \bigwedge_{\{\Theta \in T \mid s'_\Theta = s'\}} \phi_\Theta.$$

It is clear that $\overline{s} \models \phi$, hence $\overline{t} \models \phi$ by $\Delta =_{PB} \Theta$. It follows that there must be a Θ^* with $\overline{t} \xrightarrow{a} \Theta^*$, $\Theta^* = \sum_{s' \in \lceil \Delta \rceil} \Delta(s') \cdot \Theta^*_{s'}$ and for each $s' \in \lceil \Delta \rceil, t' \in \lceil \Theta^*_{s'} \rceil$ we have $\overline{t'} \models \bigwedge_{\{\Theta \in T \mid s'_\Theta = s'\}} \phi_\Theta$. This means that $\Theta^* \notin T$ and hence for each $s' \in \lceil \Delta \rceil, t' \in \lceil \Theta^*_{s'} \rceil$ we have $\overline{s'} =_{PB} \overline{t'}$, i.e. $s' \mathcal{R} t'$. Consequently, we obtain $\Delta \overline{\mathcal{R}} \Theta^*$. By symmetry all transitions of t can be matched up by transitions of s. □

4.3 Characteristic formulae

4.3.1 Probabilistic modal mu-calculus

Let *Var* be a countable set of variables. We define a set $\mathcal{L}_\mu$ of modal formulae in positive normal form given by the following grammar:

$$\phi := \langle a \rangle \phi \mid [a]\phi \mid \bigwedge_{i \in I} \phi_i \mid \bigvee_{i \in I} \phi_i \mid \bigoplus_{i \in I} p_i \cdot \phi_i \mid X \mid \mu X.\phi \mid \nu X.\phi$$

where $a \in Act$, I is an index set and $\sum_{i \in I} p_i = 1$. As usual, we have $\bigwedge_{i \in \emptyset} \phi_i = \top$ and $\bigvee_{i \in \emptyset} \phi_i = \bot$.

The two fixpoint operators μX and νX bind the respective variable X. We apply the usual terminoloty of free and bound variables in a formula and write $fv(\phi)$ for the set of free variables in ϕ.

We use *environments*, which binds free variables to sets of distributions, in order to give semantics to formulae. Let

$$\mathsf{Env} = \{\, \rho \mid \rho : Var \to \mathcal{P}(\mathcal{D}(S)) \,\}$$

be the set of all environments and ranged over by ρ. For a set $V \subseteq \mathcal{D}(S)$ and a variable $X \in Var$, we write $\rho[X \mapsto V]$ for the environment that maps X to V and Y to $\rho(Y)$ for all $Y \neq X$.

The semantics of a formula ϕ can be given as the set of distributions satisfying it. This leads to a semantic functional $[\![\ ]\!] : \mathcal{L}_\mu \to \mathsf{Env} \to \mathcal{P}(\mathcal{D}(S))$ defined inductively in Figure 4.2. As the meaning of a closed formula ϕ does not depend on the environment, we write $[\![\phi]\!]$ for $[\![\phi]\!]_\rho$ where ρ is an arbitrary environment.

The semantics of probabilistic modal mu-calculus (pMu) is the same as that of the modal mu-calculus [Koz83] except that distributions are taking the roles of states. The characterisation of *least fixpoint formula* $\mu X.\phi$ and *greatest fixpoint formula* $\nu X.\phi$ follows from the well-known Knaster-Tarski fixpoint theorem [Tar55].

We shall consider (closed) *equation systems* of formulae of the form

$$\begin{aligned} E : X_1 &= \phi_1 \\ &\vdots \\ X_n &= \phi_n \end{aligned}$$

$$
\begin{array}{rcl}
\left[\!\left[\top\right]\!\right]_\rho & = & \mathcal{D}(S) \\
\left[\!\left[\bot\right]\!\right]_\rho & = & \emptyset \\
\left[\!\left[\bigwedge_{i\in I}\phi_i\right]\!\right]_\rho & = & \bigcap_{i\in I}\left[\!\left[\phi_i\right]\!\right]_\rho \\
\left[\!\left[\bigvee_{i\in I}\phi_i\right]\!\right]_\rho & = & \bigcup_{i\in I}\left[\!\left[\phi_i\right]\!\right]_\rho \\
\left[\!\left[\bigoplus_{i\in I} p_i\cdot\phi_i\right]\!\right]_\rho & = & \{\,\Delta\in\mathcal{D}(S)\mid \Delta=\bigoplus_{i\in I}p_i\cdot\Delta_i \ \wedge\ \forall i\in I,\forall t\in\lceil\Delta_i\rceil : \overline{t}\in\left[\!\left[\phi_i\right]\!\right]_\rho\,\} \\
\left[\!\left[\langle a\rangle\phi\right]\!\right]_\rho & = & \{\,\Delta\in\mathcal{D}(S)\mid \exists\Delta' : \Delta\xrightarrow{a}\Delta' \ \wedge\ \Delta'\in\left[\!\left[\phi\right]\!\right]_\rho\,\} \\
\left[\!\left[[a]\phi\right]\!\right]_\rho & = & \{\,\Delta\in\mathcal{D}(S)\mid \forall\Delta' : \Delta\xrightarrow{a}\Delta' \ \Rightarrow\ \Delta'\in\left[\!\left[\phi\right]\!\right]_\rho\,\} \\
\left[\!\left[X\right]\!\right]_\rho & = & \rho(X) \\
\left[\!\left[\mu X.\phi\right]\!\right]_\rho & = & \bigcap\{\,V\subseteq\mathcal{D}(S)\mid \left[\!\left[\phi\right]\!\right]_{\rho[X\mapsto V]}\subseteq V\,\} \\
\left[\!\left[\nu X.\phi\right]\!\right]_\rho & = & \bigcup\{\,V\subseteq\mathcal{D}(S)\mid \left[\!\left[\phi\right]\!\right]_{\rho[X\mapsto V]}\supseteq V\,\}
\end{array}
$$

Figure 4.2: Semantics of probabilistic modal mu-calculus

where $X_1, ..., X_n$ are mutually distinct variables and $\phi_1, ..., \phi_n$ are formulae haveing at most $X_1, ..., X_n$ as free variables. Here E can be viewed as a function $E : Var \to \mathcal{L}_\mu$ defined by $E(X_i) = \phi_i$ for $i = 1, ..., n$ and $E(Y) = Y$ for other variables $Y \in Var$.

An environment ρ is a *solution* of an equation system E if $\forall i : \rho(X_i) = \left[\!\left[\phi_i\right]\!\right]_\rho$. The existence of solutions for an equation system can be seen from the following arguments. The set Env, which includes all candidates for solutions, together with the partial order $\leq$ defined by

$$\rho \leq \rho' \text{ iff } \forall X \in Var : \rho(X) \subseteq \rho'(X)$$

forms a complete lattice. The *equation functional* $\mathcal{E} : \mathsf{Env} \to \mathsf{Env}$ given in the λ-calculus notation by

$$\mathcal{E} := \lambda\rho.\lambda X.\left[\!\left[E(X)\right]\!\right]_\rho$$

is monotonic. Thus, the Knaster-Tarski fixpoint theorem guarantees existence of solutions, and the largest solution

$$\rho_E := \bigsqcup\{\,\rho \mid \rho \leq \mathcal{E}(\rho)\,\}$$

4.3.2 Characteristic equation systems

As studied in [SI94], the behaviour of a process can be characterised by an equation system of modal formulae. Below we show that this idea also applies in the probabilistic setting.

Strong bisimulation

Definition 4.11 *Given a finite state pLTS, its* characteristic equation system *for strong bisimulation consists of one equation for each state $s_1, ..., s_n \in S$.*

$$
\begin{array}{rcl}
E : X_{s_1} & = & \phi_{s_1} \\
 & \vdots & \\
X_{s_n} & = & \phi_{s_n}
\end{array}
$$

where

$$\phi_s := \Big(\bigwedge_{s\xrightarrow{a}\Delta}\langle a\rangle X_\Delta\Big)\wedge\Big(\bigwedge_{a\in Act_\tau}[a]\bigvee_{\overline{s}\xrightarrow{a}\Delta}X_\Delta\Big) \tag{4.11}$$

with $X_\Delta := \bigoplus_{s\in\lceil\Delta\rceil}\Delta(s)\cdot X_s$.

Theorem 4.12 *Suppose E is a characteristic equation system for strong bisimulation. Then $s \sim t$ iff $\overline{t} \in \rho_E(X_s)$.*

Proof: ($\Leftarrow$) Let $\mathcal{R} = \{ (s,t) \mid \overline{t} \in \rho_E(X_s) \}$. We first show that

$$\Theta \in [\![X_\Delta]\!]_{\rho_E} \text{ implies } \Delta \; \overline{\mathcal{R}} \; \Theta. \tag{4.12}$$

Let $\Delta = \bigoplus_{i\in I} p_i \cdot \overline{s_i}$, then $X_\Delta = \bigoplus_{i\in I} p_i \cdot X_{s_i}$. Suppose $\Theta \in [\![X_\Delta]\!]_{\rho_E}$. We have that $\Theta = \bigoplus_{i\in I} p_i \cdot \Theta_i$ and, for all $i \in I$ and $t \in \lceil \Delta_i \rceil$, that $\overline{t} \in [\![X_{s_i}]\!]_{\rho_E}$, i.e. $s_i \; \mathcal{R} \; t$. It follows that $\overline{s_i} \; \overline{\mathcal{R}} \; \Theta_i$ and thus $\Delta \; \overline{\mathcal{R}} \; \Theta$.

Now we show that $\mathcal{R}$ is a bisimulation.

1. Suppose $s \; \mathcal{R} \; t$ and $s \xrightarrow{a} \Delta$. Then $\overline{t} \in \rho_E(X_s) = [\![\phi_s]\!]_{\rho_E}$. It follows from (4.11) that $\overline{t} \in [\![\langle a \rangle X_\Delta]\!]_{\rho_E}$. So there exists some Θ such that $\overline{t} \xrightarrow{a} \Theta$ and $\Theta \in [\![X_\Delta]\!]_{\rho_E}$. Now we apply (4.12).
2. Suppose $s \; \mathcal{R} \; t$ and $t \xrightarrow{a} \Theta$. Then $\overline{t} \in \rho_E(X_s) = [\![\phi_s]\!]_{\rho_E}$. It follows from (4.11) that $\overline{t} \in [\![[a] \bigvee_{\overline{s} \xrightarrow{a} \Delta} X_\Delta]\!]$. Notice that it must be the case that $\overline{s} \xrightarrow{a}$, otherwise, $\overline{t} \in [\![[a]\bot]\!]_{\rho_E}$ and thus $t \not\xrightarrow{a}$, in contradiction with the assumption $t \xrightarrow{a} \Theta$. Therefore, $\Theta \in [\![\bigvee_{\overline{s} \xrightarrow{a} \Delta} X_\Delta]\!]_{\rho_E}$, which implies $\Theta \in [\![X_\Delta]\!]_{\rho_E}$ for some Δ with $\overline{s} \xrightarrow{a} \Delta$. Now we apply (4.12).

($\Rightarrow$) We define the environment $\rho_\sim$ by

$$\rho_\sim(X_s) := \{ \overline{t} \mid s \sim t \}.$$

It sufficies to show that $\rho_\sim$ is a post-fixpoint of $\mathcal{E}$, i.e.

$$\rho_\sim \leq \mathcal{E}(\rho_\sim) \tag{4.13}$$

because in that case we have $\rho_\sim \leq \rho_E$, thus $s \sim t$ implies $\overline{t} \in \rho_\sim(X_s)$ implies $\overline{t} \in \rho_E(X_s)$.

We first show that

$$\Delta \eqsim \Theta \text{ implies } \Theta \in [\![X_\Delta]\!]_{\rho_\sim}. \tag{4.14}$$

Suppose $\Delta \eqsim \Theta$, we have that (i) $\Delta = \bigoplus_{i\in I} p_i \cdot \overline{s_i}$, (ii) $\Theta = \bigoplus_{i\in I} p_i \cdot \overline{t_i}$, (iii) $s_i \sim t_i$ for all $i \in I$. We know from (iii) that $\overline{t_i} \in [\![X_{s_i}]\!]_{\rho_\sim}$. Using (ii) we have that $\Theta \in [\![\bigoplus_{i\in I} p_i \cdot X_{s_i}]\!]_{\rho_\sim}$. Using (i) we obtain $\Theta \in [\![X_\Delta]\!]_{\rho_\sim}$.

Now we are in a position to show (4.13). Suppose $\overline{t} \in \rho_\sim(X_s)$. We must prove that $\overline{t} \in [\![\phi_s]\!]_{\rho_\sim}$, i.e.

$$\overline{t} \in \Big(\bigcap_{s \xrightarrow{a} \Delta} [\![\langle a \rangle X_\Delta]\!]_{\rho_\sim} \Big) \cap \Big(\bigcap_{a \in Act_\tau} [\![[a] \bigvee_{\overline{s} \xrightarrow{a} \Delta} X_\Delta]\!]_{\rho_\sim} \Big)$$

by (4.11). This can be done by showing that $\overline{t}$ belongs to each of the two parts of this intersection.

1. In the first case, we assume that $s \xrightarrow{a} \Delta$. Since $s \sim t$, there exists some Θ such that $\overline{t} \xrightarrow{a} \Theta$ and $\Delta \eqsim \Theta$. By (4.14), we get $\Theta \in [\![X_\Delta]\!]_{\rho_\sim}$. It follows that $\overline{t} \in [\![\langle a \rangle X_\Delta]\!]_{\rho_\sim}$.
2. In the second case, there are two possibilities.
   - $s \not\xrightarrow{a}$. Then $\bigvee_{s \xrightarrow{a} \Delta} X_\Delta = \bot$. Since $s \sim t$, we have $t \not\xrightarrow{a}$, thus $\overline{t} \in [\![[a]\bot]\!]_{\rho_\sim}$.
   - $s \xrightarrow{a}$. If $\overline{t} \xrightarrow{a} \Theta$, then by $s \sim t$ there exists some Δ such that we have $\overline{s} \xrightarrow{a} \Delta$ and $\Delta \eqsim \Theta$. By (4.14), we get $\Theta \in [\![X_\Delta]\!]_{\rho_\sim}$. As a consequence, $\overline{t} \in [\![[a] \bigvee_{\overline{s} \xrightarrow{a} \Delta} X_\Delta]\!]_{\rho_\sim}$.

□

Strong simulation

Characteristic equation systems for strong simulation are defined as in Definition 4.11 except that we drop the second part of the intersection in (4.11), so ϕ_s takes the following form

$$\phi_s := \bigwedge_{s \xrightarrow{a} \Delta} \langle a \rangle X_\Delta \tag{4.15}$$

With this modification, we have the following theorem, which can be shown by using the ideas in the proof of Theorem 4.12, and with fewer cases to analyse.

Theorem 4.13 *Suppose E is a characteristic equation system for strong simulation. Then $s \prec t$ iff $\overline{t} \in \rho_E(X_s)$.*

1. Rule 1: $E \to F$
2. Rule 2: $E \to G$
3. Rule 3: $E \to H$ if $X_n \notin fv(\phi_1, ..., \phi_n)$

$$
\begin{array}{rcl}
E : X_1 & = & \phi_1 \\
 & \vdots & \\
X_{n-1} & = & \phi_{n-1} \\
X_n & = & \phi_n
\end{array}
\qquad
\begin{array}{rcl}
F : X_1 & = & \phi_1 \\
 & \vdots & \\
X_{n-1} & = & \phi_{n-1} \\
X_n & = & \nu X_n.\phi_n
\end{array}
\qquad
\begin{array}{rcl}
G : X_1 & = & \phi_1[\phi_n/X_n] \\
 & \vdots & \\
X_{n-1} & = & \phi_{n-1}[\phi_n/X_n] \\
X_n & = & \phi_n
\end{array}
\qquad
\begin{array}{rcl}
H : X_1 & = & \phi_1 \\
 & \vdots & \\
X_{n-1} & = & \phi_{n-1}
\end{array}
$$

Figure 4.3: Transformation rules

4.3.3 Characteristic formulae

So far we know how to construct characteristic equation systems for a finite state pLTS. As introduced in [MO98], the three transformation rules in Figure 4.3 can be used to obtain from an equation system E a formula whose interpretation coincides with the interpretation of X_1 in the greatest solution of E. The formula thus obtained from a characteristic equation system is called a *characteristic formula*.

Theorem 4.14 *Given a characteristic equation system E, there is a characteristic formula ϕ_s such that $\rho_E(X_s) = [\![\phi_s]\!]$ for any state s.*

The above theorem, together with the results in Section 4.3.2, leads to the following corollary.

Corollary 4.15 *For each state s in a finite state pLTS,*

1. *there is a characteristic formula $\phi_s^{\sim}$ such that $s \sim t$ iff $\overline{t} \in [\![\phi_s^{\sim}]\!]$;*
2. *there is a characteristic formula $\phi_s^{\prec}$ such that $s \prec t$ iff $\overline{t} \in [\![\phi_s^{\prec}]\!]$.*

4.4 Metric analogue of bisimulation

In the bisimulation game probabilities are treated as labels since they are matched only when they are identical. One may argue that this does not provide a robust relation: Processes that differ for a very small probability, for instance, would be considered just as different as processes that perform completely different actions. This is particularly relevant to security systems where specifications can be given as perfect, but impractical processes and other, practical processes are considered safe if they only differ from the specification with a negligible probability.

To find a more flexible way to differentiate processes, researchers in this area have borrowed from pure mathematics the notion of metric[1]. A metric is defined as a function that associates a set distance with a pair of elements. Whereas topologists use metrics as a tool to study continuity and convergence, we will use them to provide a measure of the difference between two processes that are not quite bisimilar.

Since different processes may behave the same, they will be given distance zero in our metric semantics. So we are more interested in pseudometrics than metrics.

In the rest of this section, we fix a finite-state pLTS $(S, Act, \longrightarrow)$ and provide the set of pseudometrics on S with the following partial order.

[1]For simplicity, in this section we use the term metric to denote both metric and pseudometric. All the results are based on pseudometrics.

Definition 4.16 *The relation $\preceq$ for the set $\mathcal{M}$ of 1-bounded pseudometrics on S is defined by*

$$m_1 \preceq m_2 \text{ if } \forall s,t : m_1(s,t) \geq m_2(s,t).$$

Here we reverse the ordering with the purpose of characterizing bisimilarity as the *greatest* fixed point (cf: Corollary 4.28).

Lemma 4.17 *$(\mathcal{M}, \preceq)$ is a complete lattice.*

Proof: The top element is given by $\forall s,t : \top(s,t) = 0$; the bottom element is given by $\bot(s,t) = c$ if $s \neq t$, 0 otherwise. Greatest lower bounds are given by $(\bigsqcap X)(s,t) = sup\{m(s,t) \mid m \in X\}$ for any $X \subseteq \mathcal{M}$. Finally, least upper bounds are given by $\bigsqcup X = \bigsqcap \{m \in \mathcal{M} \mid \forall m' \in X : m' \preceq m\}$. □

In order to define the notion of state-metrics (which will correspond to bisimulations) and the monotone transformation on metrics, we need to associate a metric with $\mathcal{D}(S)$. Here we give a definition based on the Kantorovich metric [Kan42] on probability measures, which has been used by van Breugel and Worrell for defining metrics on fully probabilistic systems [vBW01a] and reactive probabilistic systems [vBW01b]; and by Desharnais *et al.* for labelled Markov chains [DJGP02] and labelled concurrent Markov chains [DJGP04], respectively.

Definition 4.18 *For each $m \in \mathcal{M}$, we lift it to be a metric $\hat{m}$ over $\mathcal{D}(S)$. Given $\Delta, \Delta' \in \mathcal{D}(S)$, we define $\hat{m}(\Delta, \Delta')$ as the solution to the following linear program:*

$$\begin{array}{ll} \textit{maximize} & \sum_{s \in S}(\Delta(s) - \Delta'(s))x_s \\ \textit{subject to} & \bullet\ \forall s \in S : 0 \leq x_s \leq 1 \\ & \bullet\ \forall s, s' \in S : x_s - x_{s'} \leq m(s,s') \end{array} \tag{4.16}$$

An alternative definition would be to scale the above $\hat{m}(\Delta, \Delta')$ by a factor $e \in (0,1]$, see van Breugel and Worrell [vBW05] for more discussions. Here we simply let $e = 1$ because all the main results obtained in this section are independent from e.

Proposition 4.19 *For each $m \in \mathcal{M}$, $\hat{m}$ is a metric over $\mathcal{D}(S)$.*

Proof: We verify that $(\mathcal{D}(S), \hat{m})$ satisfies the definition of pseudometric space.

1. It is clear that $\hat{m}(\Delta, \Delta) = 0$.

2. We observe that

$$\begin{array}{rcl} \sum_{s \in S}(\Delta(s) - \Delta'(s))x_s & = & \sum_{s \in S}(\Delta'(s) - \Delta(s))(1 - x_s) + \sum_{s \in S}\Delta(s) - \sum_{s \in S}\Delta'(s) \\ & = & \sum_{s \in S}(\Delta'(s) - \Delta(s))(1 - x_s) \end{array}$$

   Now $x'_s = 1 - x_s$ also satisfy the constraints on x_s in Definition 4.18, hence the symmetry of $\hat{m}$ can be shown.

3. Let $\Delta_1, \Delta_2, \Delta_3 \in \mathcal{D}(S)$, we have

$$\sum_{s \in S}(\Delta_1(s) - \Delta_3(s))x_s = \sum_{s \in S}(\Delta_1(s) - \Delta_2(s))x_s + \sum_{s \in S}(\Delta_2(s) - \Delta_3(s))x_s.$$

   By taking the maximum over the x_s for the left hand side, we obtain

$$\hat{m}(\Delta_1, \Delta_3) \leq \hat{m}(\Delta_1, \Delta_2) + \hat{m}(\Delta_2, \Delta_3).$$

□

Definition 4.20 *$m \in \mathcal{M}$ is a* state-metric *if, for all $\epsilon \in [0,1)$, $m(s,t) \leq \epsilon$ implies:*

- *if $s \xrightarrow{a} \Delta$ then there exists some Δ' such that $t \xrightarrow{a} \Delta'$ and $\hat{m}(\Delta, \Delta') \leq \epsilon$.*

Note that if m is a state-metric then it is also a metric. By $m(s,t) \leq \epsilon$ we have $m(t,s) \leq \epsilon$, which implies

- if $t \xrightarrow{a} \Delta'$ then there exists some Δ such that $s \xrightarrow{a} \Delta$ and $\hat{m}(\Delta', \Delta) \leq \epsilon$.

In the above definition, we prohibit ϵ to be 1 because we use 1 to represent the distance between any two incomparable states including the case where one state may perform a transition and the other may not.

The greatest state-metric is defined as

$$m_{max} = \bigsqcup \{m \in \mathcal{M} \mid m \text{ is a state-metric}\}.$$

When compared with Proposition 2.11, it turns out that state-metrics correspond to bisimulations and the greatest state-metric corresponds to bisimilarity. To make the analogy closer, in what follows we will characterize m_{max} as a fixed point of a suitable monotone function on $\mathcal{M}$. First we recall the definition of Hausdorff distance.

Definition 4.21 *Given a 1-bounded metric d on Z, the* Hausdorff distance *between two subsets X, Y of Z is defined as follows:*

$$H_d(X,Y) = max\{sup_{x \in X} inf_{y \in Y} d(x,y), sup_{y \in Y} inf_{x \in X} d(y,x)\}$$

where $inf\ \emptyset = 1$ and $sup\ \emptyset = 0$.

Next we define a function F on $\mathcal{M}$ by using the Hausdorff distance.

Definition 4.22 *Let $der(s,a) = \{\Delta \mid s \xrightarrow{a} \Delta\}$. $F(m)$ is a pseudometric given by:*

$$F(m)(s,t) = max_{a \in Act}\{H_{\hat{m}}(der(s,a), der(t,a))\}.$$

Thus we have the following property.

Lemma 4.23 *For all $\epsilon \in [0,1)$, $F(m)(s,t) \leq \epsilon$ if and only if:*

- *if $s \xrightarrow{a} \Delta$ then there exists some Δ' such that $t \xrightarrow{a} \Delta'$ and $\hat{m}(\Delta, \Delta') \leq \epsilon$;*
- *if $t \xrightarrow{a} \Delta'$ then there exists some Δ such that $s \xrightarrow{a} \Delta$ and $\hat{m}(\Delta', \Delta) \leq \epsilon$.*

□

The above lemma can be proved by directly checking the definition of F, as can the next lemma.

Lemma 4.24 *m is a state-metric iff $m \preceq F(m)$.* □

Consequently we have the following characterisation:

$$m_{max} = \bigsqcup \{m \in \mathcal{M} \mid m \preceq F(m)\}.$$

Lemma 4.25 *F is monotone on $\mathcal{M}$.* □

Because of Lemma 4.17 and 4.25, we can apply Theorem 1.4, which tells us that m_{max} is the greatest fixed point of F. Furthermore, by Lemma 4.24 we know that m_{max} is indeed a state-metric, and it is the greatest state-metric.

In addition, if our pLTS is image-finite, i.e. for all $a \in Act, s \in S$ the set $der(s,a)$ is finite, the closure ordinal of F is ω. Therefore one can proceed in a standard way to show that

$$m_{max} = \bigsqcap \{F^i(\top) \mid i \in \mathbb{N}\}$$

where $\top$ is the top metric in $\mathcal{M}$ and $F^0(\top) = \top$.

Lemma 4.26 *For image-finite pLTSs, the closure ordinal of F is ω.*

Proof: Let $m_{max}(s,t) \leq \epsilon$. Let $s \xrightarrow{a} \Delta$. For each $m_i = F^i(\top)$ there is a Θ_i such that $t \xrightarrow{a} \Theta_i$ and $\hat{m}_i(\Delta, \Theta) \leq \epsilon$. Since the pLTSs are image-finite, there is a Θ such that for all but finitely many i, $t \xrightarrow{a} \Theta$ and $\hat{m}_i(\Delta, \Theta) \leq \epsilon$. □

We now show the correspondence between our state-metrics and bisimulations.

Theorem 4.27 *Given a binary relation $\mathcal{R}$ and a pseudometric $m \in \mathcal{M}$ such that*

$$m(s,t) = \begin{cases} 0 & \text{if } s\mathcal{R}t \\ 1 & \text{otherwise.} \end{cases}$$

Then $\mathcal{R}$ is a bisimulation iff m is a state-metric.

Proof: Given two distributions Δ, Δ' over S, let us consider how to compute $\hat{m}(\Delta, \Delta')$ if $\mathcal{R}$ is an equivalence relation. Since S is finite, we may assume that $V_1, ..., V_n \in S/\mathcal{R}$ are all the equivalence classes of S under $\mathcal{R}$. If $s, t \in V_i$ for some $i \in 1..n$, then $m(s,t) = 0$, which implies $x_s = x_t$ by the second constraint of (4.16). So for each $i \in 1..n$ there exists some x_i such that $x_i = x_s$ for all $s \in V_i$. Thus, some summands of (4.16) can be grouped together and we have the following linear program:

$$\sum_{i \in 1..n} (\Delta(V_i) - \Delta'(V_i))x_i \tag{4.17}$$

with the constraint $x_i - x_j \leq 1$ for any $i, j \in 1..n$ with $i \neq j$. Briefly speaking, if $\mathcal{R}$ is an equivalence relation then $\hat{m}(\Delta, \Delta')$ is obtained by maximizing the linear program (4.17).

($\Rightarrow$) Suppose $\mathcal{R}$ is a bisimulation and $m(s,t) = 0$. By assumption $\mathcal{R}$ is clearly an equivalence relation. By the definition of m we have $s\mathcal{R}t$. If $s \xrightarrow{a} \Delta$ then $t \xrightarrow{a} \Delta'$ for some Δ' such that $\Delta \; \overline{\mathcal{R}} \; \Delta'$. To show that m is a state-metric it suffices to prove $m(\Delta, \Delta') = 0$. We know from $\Delta \; \overline{\mathcal{R}} \; \Delta'$ and Proposition 4.4 (2) that $\Delta(V_i) = \Delta'(V_i)$, for each $i \in 1..n$. It follows that (4.17) is maximized to be 0, thus $m(\Delta, \Delta') = 0$.

($\Leftarrow$) Suppose m is a state-metric and has the relation as defined in the hypothesis. Notice that $\mathcal{R}$ is an equivalence relation. We show that it is a bisimulation. Suppose $s\mathcal{R}t$, which means $m(s,t) = 0$. If $s \xrightarrow{a} \Delta$ then $t \xrightarrow{a} \Delta'$ for some Δ' such that $m(\Delta, \Delta') = 0$. To ensure that $m(\Delta, \Delta') = 0$, in (4.17) the following two conditions must be satisfied.

1. No coefficient is positive. Otherwise, if $\Delta(V_i) - \Delta'(V_i) > 0$ then (4.17) would be maximized to a value not less than $(\Delta(V_i) - \Delta'(V_i))$, which is greater than 0.

2. It is not the case that at least one coefficient is negative and the other coefficients are either negative or 0. Otherwise, by summing up all the coefficients, we would get

   $$\Delta(S) - \Delta'(S) < 0$$

   which contradicts the assumption that Δ and Δ' are distributions over S.

Therefore the only possibility is that all coefficients in (4.17) are 0, i.e., $\Delta(V_i) = \Delta'(V_i)$ for any equivalence class $V_i \in S/\mathcal{R}$. It follows from Proposition 4.4 (2) that $\Delta \; \overline{\mathcal{R}} \; \Delta'$. So we have shown that $\mathcal{R}$ is indeed a bisimulation. □

Corollary 4.28 *$s \sim_p t$ iff $m_{max}(s,t) = 0$.*

Proof: ($\Rightarrow$) Since $\sim_p$ is a bisimulation, by Theorem 4.27 there exists some state-metric m such that $s \sim_p t$ iff $m(s,t) = 0$. By the definition of m_{max} we have $m \preceq m_{max}$. Therefore $m_{max}(s,t) \leq m(s,t) = 0$.

($\Leftarrow$) From m_{max} we construct a pseudometric m as follows.

$$m(s,t) = \begin{cases} 0 & \text{if } m_{max}(s,t) = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Since m_{max} is a state-metric, it is easy to see that m is also a state-metric. Now we construct a binary relation $\mathcal{R}$ such that $\forall s, s' : s\mathcal{R}s'$ iff $m(s, s') = 0$. If follows from Theorem 4.27 that $\mathcal{R}$ is a bisimulation. If $m_{max}(s, t) = 0$, then $m(s, t) = 0$ and thus $s\mathcal{R}t$. Therefore we have the required result $s \sim_p t$ because $\sim_p$ is the largest bisimulation. □

4.5 Equivalence checking

In this section we consider a probabilistic extension of strong simulation and bisimulation.

Definition 4.29 *A relation $\mathcal{R} \subseteq S \times S$ is a* strong simulation *if $s\ \mathcal{R}\ t$ implies*

- *if $s \xrightarrow{a} \Delta$ then there exists some Θ such that $t \xrightarrow{a} \Theta$ and $\Delta\ \overline{\mathcal{R}}\ \Theta$.*

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong simulations, then $\mathcal{R}$ is a strong bisimulation. *We $s \sim t$ if there exists a strong bisimulation $\mathcal{R}$ with $s\ \mathcal{R}\ t$.*

We present two algorithms for computing bisimulation and simulation [BEMC00].

4.5.1 Computing strong bisimulation

We will give a partition-refinement algorithm for pLTSs. Before that we briefly sketch how the partitioning technique presented in Section 2.7.1 can be modified for reactive pLTSs and explain why this method fails for general pLTSs.

The partitioning technique for reactive pLTSs Let $\langle S, A, \rightarrow\rangle$ be a reactive pLTS. For any $a \in A$ and $B \subseteq S$, we define the equivalence relation $\sim_{(a,B)}$ by letting $s \sim_{(a,B)} t$ if $s \xrightarrow{a} \Delta$ and $t \xrightarrow{a} \Theta$ with $\Delta(B) = \Theta(B)$. We still use the schema shown in Figure 2.4 and the refinement operator in (2.7), but change the splitting procedure as follows

$$\mathbf{Split}(B, a, \mathcal{B}) = \bigcup_{C \in \mathcal{B}} \mathbf{Split}(B, a, C) \text{ where } \mathbf{Split}(B, a, C) = C/\sim_{(a,B)} . \tag{4.18}$$

An implementation of the algorithm using some tricks on data structures yields the following complexity result.

Theorem 4.30 *The bisimulation equivalence classes of a reactive pLTS with n states and m transitions can be computed in time $\mathcal{O}(mn \log n)$ and space $\mathcal{O}(mn)$.*

The splitter technique in (4.18) does not work for general pLTSs when we use the obvious modification of the equivalence relation $\sim_{(a,B)}$ where $s \sim_{(a,B)} t$ iff for any transition $s \xrightarrow{a} \Delta$ there is a transition $t \xrightarrow{a} \Theta$ with $\Delta(B) = \Theta(B)$ and vice versa.

Example 4.31 *Consider the pLTS described in Figure 4.4, we have $s \not\sim s'$ because the transition $s \xrightarrow{a} \frac{1}{2}\overline{t} + \frac{1}{2}\overline{u}$ cannot be matched by any transition from s'. However, s and s' cannot be distinguished by using the above partitioning technique. The problem is that after one round of refinement, we obtain the blocks*

$$\{s, s'\}, \{t\}, \{u\}, \{v\}, \{w\}$$

and then no further refinement can split the block $\{s, s'\}$.

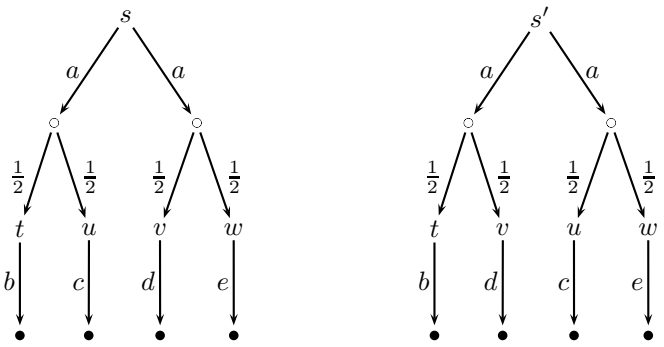


Figure 4.4: $s \not\sim s'$

The partitioning technique for pLTSs To compute bisimulation equivalence classes in general pLTSs, we can keep the schema sketched in Figure 2.4 but use two partitions: a partition $\mathcal{B}$ for states and a partition $\mathcal{M}$ for transitions. By a *transition partition* we mean a set $\mathcal{M}$ consisting of pairs (a, M) where $a \in A$ and $M \subseteq M_a$ with $M_a = \bigcup_{s \in S} \{\Delta \mid s \xrightarrow{a} \Delta\}$ such that, for any action a, the set $\{M \mid (a, M) \in \mathcal{M}\}$ is a partition of the set M_a.

The algorithm works as follows. We skip the first refinement step and start with the state partition

$$\mathcal{B}_{init} = S/\sim_A \text{ where } s \sim_A t \text{ iff } \{a \mid s \xrightarrow{a}\} = \{a \mid t \xrightarrow{a}\}$$

that identifies those states that can perform the same actions immediately. The initial transition partition

$$\mathcal{M}_{init} = \{(a, M_a) \mid a \in A\}$$

identifies all transitions with the same label. In each iteration, we try to refine the state partition $\mathcal{B}$ according to an equivalence class (a, M) of $\mathcal{M}$ or the transition partition $\mathcal{M}$ according to a block $B \in \mathcal{B}$. The refinement of $\mathcal{B}$ by (a, M) is done by the operation $\mathbf{Split}(M, a, \mathcal{B})$ that divides each block B of $\mathcal{B}$ into two subblocks $B_{(a,M)} = \{s \in B \mid s \xrightarrow{a} M\}$ and $B \backslash B_{(a,M)}$. In other words,

$$\mathbf{Split}(M, a, \mathcal{B}) \;=\; \bigcup_{B \in \mathcal{B}} \mathbf{Split}(M, a, B)$$

where $\mathbf{Split}(M, a, B) = \{B_{(a,M)}, B \backslash B_{(a,M)}\}$ and $B_{(a,M)} = \{s \in B \mid s \xrightarrow{a} M\}$. The refinement of $\mathcal{M}$ by B is done by the operation $\mathbf{Split}(B, \mathcal{M})$ that divides any block $(a, M) \in \mathcal{M}$ by the subblocks $(a, M_1), ..., (a, M_n)$ where $\{M_1, ..., M_n\} = M/\sim_B$ and $\Delta \sim_B \Theta$ iff $\Delta(B) = \Theta(B)$. Formally,

$$\mathbf{Split}(B, \mathcal{M}) \;=\; \bigcup_{(a,M) \in \mathcal{M}} \mathbf{Split}(B, (a, M))$$

where $\mathbf{Split}(B, (a, M)) = \{(a, M') \mid M' \in M/\sim_B\}$. If no further refinement of $\mathcal{B}$ and $\mathcal{M}$ is possible then we have $\mathcal{B} = S/\sim$. The algorithm is sketched in Figure 4.5. See [BEMC00] for the correctness proof of the algorithm and the suitable data structures used to obtain the following complexity result.

Theorem 4.32 *The bisimulation equivalence classes of a pLTS with n states and m transitions can be decided in time $\mathcal{O}(mn(\log m + \log n))$ and space $\mathcal{O}(mn)$.*

$\mathcal{B} := S/\sim_A$ where $s \sim_A t$ iff $\{a \mid s \xrightarrow{a}\} = \{a \mid t \xrightarrow{a}\}$
$\mathcal{M} := \{(a, M_a) \mid a \in A\}$ where $M_a = \bigcup_{s \in S}\{\Delta \mid s \xrightarrow{a} \Delta\}$
As long as $\mathcal{B}$ or $\mathcal{M}$ can be modified perform one of the following steps:
- either choose some $B \in \mathcal{B}$ and put $\mathcal{M} := \mathbf{Split}(B, \mathcal{M})$
- or choose some $(a, M) \in \mathcal{M}$ and put $\mathcal{B} := \mathbf{Split}(M, a, \mathcal{B})$

Return $\mathcal{B}$

Figure 4.5: The algorithm for computing bisimulation equivalence classes in pLTSs

$\mathcal{R} := S \times S$
While there exists $(s, t) \in \mathcal{R}$ with $s \not\prec_{\mathcal{R}} t$ do
$\mathcal{R} := \mathcal{R} \backslash \{(s, t)\}$
Return $\mathcal{R}$

Figure 4.6: Schema for computing strong simulation

4.5.2 Computing strong simulation

The key idea of the algorithm for computing strong simulation is as in the non-probabilistic case [HHK95] (see Figure 4.6). We start with the trivial relation $\mathcal{R} = S \times S$ and successively remove those pairs (s, s') from $\mathcal{R}$ where $s \not\prec_{\mathcal{R}} t$. Intuitively, the condition $s \not\prec_{\mathcal{R}} t$ says that s has a transition that cannot be matched by a transition of t with respect to the current relation $\mathcal{R}$. Formally, we define $s \prec_{\mathcal{R}} t$ iff for each transition $s \xrightarrow{a} \Delta$ there exists $t \xrightarrow{a} \Theta$ such that $\Delta \,\overline{\mathcal{R}}\, \Theta$.

In contrast to the method for non-probabilistic LTSs [HHK95], the algorithm below relies on an explicit test for the condition $s \prec_{\mathcal{R}} t$ with a network-based technique for testing if $\Delta \,\overline{\mathcal{R}}\, \Theta$.

Networks We briefly recall the basic definitions of networks. More details can be found in e.g. [Eve79]. A *network* is a tuple $\mathcal{N} = (N, E, \bot, \top, c)$ where (N, E) is a finite directed graph (i.e. N is a set of nodes and $E \subseteq N \times N$ is a set of edges) with two special nodes $\bot$ (the *source*) and $\top$ (the *sink*) and a *capability* c, i.e. a function that assigns to each edge $(v, w) \in E$ a non-negative number $c(v, w)$. A *flow function* f for $\mathcal{N}$ is a function that assigns to edge e a real number $f(e)$ such that

- $0 \leq f(e) \leq c(e)$ for all edges e.
- Let $in(v)$ be the set of incoming edges to node v and $out(v)$ the set of outgoing edges from node v. Then, for each node $v \in N\backslash\{\bot, \top\}$,

$$\sum_{e \in in(v)} f(e) \;=\; \sum_{e \in out(v)} f(e).$$

The *flow* $F(f)$ of f is given by

$$F(f) \;=\; \sum_{e \in out(\bot)} f(e) - \sum_{e \in in(\bot)} f(e).$$

The *maximum flow* in $\mathcal{N}$ is the supremum (maximum) over the flows $F(f)$, where f is a flow function in $\mathcal{N}$.

The test whether $\Delta \,\overline{\mathcal{R}}\, \Theta$ We will see that the question whether $\Delta \,\overline{\mathcal{R}}\, \Theta$ can be reduced to a maximum flow problem in a suitably chosen network. Suppose $\mathcal{R} \subseteq S \times S$ and $\Delta, \Theta \in \mathcal{D}(S)$. Let $S' = \{s' \mid s \in S\}$ where s' are pairwise distinct new states, i.e. $s' \in S$ for all $s \in S$. We create two states $\bot$ and $\top$ not contained in $S \cup S'$ with $\bot \neq \top$. We associate with the pair (Δ, Θ) the following network $\mathcal{N}(\Delta, \Theta, \mathcal{R})$.

Input: A nonempty finite set S, distributions $\Delta, \Theta \in \mathcal{D}(S)$ and $\mathcal{R} \subseteq S \times S$
Output: If $\Delta \overline{\mathcal{R}} \Theta$ then "yes" else "no"
Method:
Construct the network $\mathcal{N}(\Delta, \Theta, \mathcal{R})$
Compute the maximum flow F in $\mathcal{N}(\Delta, \Theta, \mathcal{R})$
If $F < 1$ then return "no" else "yes".

Figure 4.7: Test whether $\Delta \overline{\mathcal{R}} \Theta$

- The nodes are $N = S \cup S' \cup \{\bot, \top\}$.
- The edges are $E = \{(s, t') \mid (s, t) \in \mathcal{R}\} \cup \{(\bot, s) \mid s \in S\} \cup \{(s', \top) \mid s \in S\}$.
- The capability c is defined by $c(\bot, s) = \Delta(s)$, $c(t', \top) = \Theta(t)$ and $c(s, t') = 1$ for all $s, t \in S$.

Lemma 4.33 *The following statements are equivalent.*

(i) There exists a weight function w for (Δ, Θ) with respect to $\mathcal{R}$.

(ii) The maximum flow in $\mathcal{N}(\Delta, \Theta, \mathcal{R})$ is 1.

Proof: (i) $\Longrightarrow$ (ii): Let w be a weight function for (Δ, Θ) with respect to $\mathcal{R}$. We define a flow function f as follows: $f(\bot, s) = \Delta(s)$, $f(t', \top) = \Theta(t)$ and $f(s, t') = w(s, t)$ for all $s, t \in S$. Then $F(f) = \sum_{s \in S} f(\bot, s) = \sum_{s \in S} \Delta(s) = 1$. So the maximum flow of $\mathcal{N}(\Delta, \Theta, \mathcal{R})$ is 1.
(ii) $\Longrightarrow$ (i): Let f be a flow function with $F(f) = 1$. Since $f(\bot, s) \leq c(\bot, s) = \Delta(s)$ and

$$\sum_{s \in S} f(\bot, s) = F(f) = 1 = \sum_{s \in S} \Delta(s)$$

it must be the case that $f(\bot, s) = \Delta(s)$ for all $s \in S$. Similarly, we get $f(t', \top) = \Theta(t)$ for all $t \in S$. Let w be the weight function defined by $w(s, t) = f(s, t')$ for all $(s, t) \in \mathcal{R}$ and $w(s, t) = 0$ if $(s, t) \notin \mathcal{R}$. We can check that

$$\sum_{t \in S} w(s, t) = \sum_{t \in S} f(s, t') = f(\bot, s) = \Delta(s)$$

and similarly, $\sum_{s \in S} w(s, t) = \Theta(t)$. So w is a weight function for (Δ, Θ) with respect to $\mathcal{R}$. □

Corollary 4.34 $\Delta \overline{\mathcal{R}} \Theta$ *iff the maximum flow in $\mathcal{N}(\Delta, \Theta, \mathcal{R})$ is* 1.

Proof: Combining Proposition 4.4(1) and Lemma 4.33. □

Corollary 4.34 provides a method for deciding whether $\Delta \overline{\mathcal{R}} \Theta$. We construct the network $\mathcal{N}(\Delta, \Theta, \mathcal{R})$ and compute the maximum flow with well-known methods, as sketched in Figure 4.7. As shown in [CHM90], computing the maximum flow in a network can be done in time $\mathcal{O}(n^3 / \log n)$ and space $\mathcal{O}(n^2)$, where n is the number of nodes in the network. So we immediately have the following result.

Lemma 4.35 *The test whether $\Delta \overline{\mathcal{R}} \Theta$ can be done in time $\mathcal{O}(n^3 / \log n)$ and space $\mathcal{O}(n^2)$.* □

Algorithm for computing simulation The algorithm for computing simulation is formulated in Figure 4.8. The key idea of the algorithm is as in Figure 4.6, but we start with the relation

$$\mathcal{R}_{init} = \{(s, t) \in S \times S \mid act(s) \subseteq act(t), s \neq t\}$$

and keep removing the pairs (Δ, Θ) for which we have detected that $\Delta \overline{\mathcal{R}} \Theta$ does not hold. Note that if Δ is not related to Θ via $\overline{\mathcal{R}}$ in the current iteration, it is also not related to Θ via $\overline{\mathcal{R}}$ in all further

$R := \{(s,t) \in S \times S \mid act(s) \subseteq act(t), s \neq t\}$, where $act(s) = \{a \mid s \xrightarrow{a}\}$
For all $(s,t) \in \mathcal{R}$ and $s \xrightarrow{a} \Delta$ do $Sim_{(s,a,\Delta)}(t) := \{\Theta \mid t \xrightarrow{a} \Theta\}$
Repeat
    $\mathcal{R}_{old} := \mathcal{R}; \mathcal{R} := \emptyset$
    For all $(s,t) \in \mathcal{R}_{old}$ do
    For all $s \xrightarrow{a} \Delta$ do
        Repeat
            Choose some $\Theta \in Sim_{(s,a,\Delta)}(t)$
            If $\Delta \overline{\mathcal{R}} \Theta$ does not hold then remove Θ from $Sim_{(s,a,\Delta)}(t)$
        until $Sim_{(s,a,\Delta)}(t) = \emptyset$ or $\Delta \overline{\mathcal{R}} \Theta$
        If $Sim_{(s,a,\Delta)}(t) \neq \emptyset$ then $\mathcal{R} := \mathcal{R} \cup \{(s,t)\}$
until $\mathcal{R}_{old} = \mathcal{R}$
Return $\mathcal{R}$

Figure 4.8: Algorithm for computing simulation in pLTSs

iterations because the relation $\mathcal{R}$ becomes smaller and smaller as the iterative procedure goes on. In the main loop, for any pair $(s,t) \in \mathcal{R}$ and transition $s \xrightarrow{a} \Delta$, we deal with the set $Sim_{(s,a,\Delta)}(t)$ that contains all distributions obtained after performing an a-transition from t that are candidates for simulating Δ. Once we detect that Δ is not related to Θ via the current relation $\mathcal{R}$, i.e. Θ cannot simulate Δ, then we remove Θ from $Sim_{(s,a,\Delta)}(t)$. To check wheter $s \prec_{\mathcal{R}} t$ for the current $\mathcal{R}$, we consider each transition $s \xrightarrow{a} \Delta$ and search in $Sim_{(s,a,\Delta)}(t)$ for a distribution Θ with $\Delta \overline{\mathcal{R}} \Theta$. If we find such a distribution then (s,t) passes the current round of testing and will go into the next round to be tested against the new relation $\mathcal{R}$. The following complexity result of the algorithm is shown in [BEMC00].

Theorem 4.36 *Computing strong simulation for a pLTS with n states and m transitions can be done in time $\mathcal{O}((mn^6 + m^2n^3)/\log n)$ and space $\mathcal{O}(mn + n^2 + m^2)$.*

4.6 Probabilistic testing semantics

4.6.1 A general testing framework

It is natural to view the semantics of processes as being determined by their ability to pass tests [DNH84, Hen88, YL92, Seg96]; processes P_1 and P_2 are deemed to be semantically equivalent unless there is a test which can distinguish them. The actual tests used typically represent the ways in which users, or indeed other processes, can interact with P_i.

Let us first set up a general testing scenario, within which this idea can be formulated. It assumes

- a set of processes $\mathcal{P}roc$
- a set of tests $\mathcal{T}$, which can be applied to processes
- a set of outcomes $\mathcal{O}$, the possible results from applying a test to a process
- a function $\mathcal{A}pply : \mathcal{T} \times \mathcal{P}roc \to \mathcal{P}^+(\mathcal{O})$, representing the possible results of applying a specific test to a specific process.

Here $\mathcal{P}^+(\mathcal{O})$ denotes the collection of non-empty subsets of $\mathcal{O}$; so the result of applying a test T to a process P, $\mathcal{A}pply(T,P)$, is in general a *set* of outcomes, representing the fact that the behaviour of processes, and indeed tests, may be nondeterministic.

Moreover, some outcomes are considered better then others; for example the application of a test may simply succeed, or it may fail, with success being better than failure. So we can assume that $\mathcal{O}$ is endowed with a partial order, in which $o_1 \leq o_2$ means that o_2 is a better outcome than o_1.

When comparing the result of applying tests to processes we need to compare subsets of $\mathcal{O}$. There are two standard approaches to make this comparison, based on viewing these sets as elements of either the Hoare or Smyth powerdomain [Hen82, AJ94] of $\mathcal{O}$. For $O_1, O_2 \in \mathcal{P}^+(\mathcal{O})$ we let

(i) $O_1 \sqsubseteq_{\mathrm{Ho}} O_2$ if for every $o_1 \in O_1$ there exists some $o_2 \in O_2$ such that $o_1 \leq o_2$

(ii) $O_1 \sqsubseteq_{\mathrm{Sm}} O_2$ if for every $o_2 \in O_2$ there exists some $o_1 \in O_1$ such that $o_1 \leq o_2$.

Using these two comparison methods we obtain two different semantic preorders for processes:

(i) For $P, Q \in \mathcal{P}roc$ let $P \sqsubseteq_{\mathrm{may}} Q$ if $\mathcal{A}pply(T, P) \sqsubseteq_{\mathrm{Ho}} \mathcal{A}pply(T, Q)$ for every test T

(ii) Similarly, let $P \sqsubseteq_{\mathrm{must}} Q$ if $\mathcal{A}pply(T, P) \sqsubseteq_{\mathrm{Sm}} \mathcal{A}pply(T, Q)$ for every test T.

We use $P \simeq_{\mathrm{may}} Q$ and $P \simeq_{\mathrm{must}} Q$ to denote the associated equivalences.

The terminology *may* and *must* refers to the following reformulation of the same idea. Let $Pass \subseteq \mathcal{O}$ be an upwards-closed subset of $\mathcal{O}$, i.e. satisfying $o' \geq o \in Pass \Rightarrow o' \in Pass$, thought of as the set of outcomes that can be regarded as *passing* a test. Then we say that a process P *may* pass a test T with an outcome in $Pass$, notation "P **may** $Pass$ T", if there is an outcome $o \in \mathcal{A}pply(P, T)$ with $o \in Pass$, and likewise P *must* pass a test T with an outcome in $Pass$, notation "P **must** $Pass$ T", if for all $o \in \mathcal{A}pply(P, T)$ one has $o \in Pass$. Now

$$P \sqsubseteq_{\mathrm{may}} Q \;\text{ iff }\; \forall T \in \mathcal{T}\, \forall Pass \in \mathcal{P}^{\uparrow}(\mathcal{O})\, (P \textbf{ may } Pass\, T \;\Rightarrow\; Q \textbf{ may } Pass\, T)$$
$$P \sqsubseteq_{\mathrm{must}} Q \;\text{ iff }\; \forall T \in \mathcal{T}\, \forall Pass \in \mathcal{P}^{\uparrow}(\mathcal{O})\, (P \textbf{ must } Pass\, T \;\Rightarrow\; Q \textbf{ must } Pass\, T)$$

where $\mathcal{P}^{\uparrow}(\mathcal{O})$ is the set of upwards-closed subsets of $\mathcal{O}$.

The original theory of testing [DNH84, Hen88] is obtained by using as the set of outcomes $\mathcal{O}$ the two-point lattice

with $\top$ representing the success of a test application, and $\bot$ failure.

However, for probabilistic processes we consider an application of a test to a process to succeed with a given probability. Thus we take as the set of outcomes the unit interval $[0, 1]$, with the standard mathematical ordering; if $0 \leq p < q \leq 1$ then succeeding with probability q is considered better than succeeding with probability p. This yields two preorders for probabilistic processes, which for convenience we rename $\sqsubseteq_{\mathrm{pmay}}$ and $\sqsubseteq_{\mathrm{pmust}}$, with the associated equivalences $\simeq_{\mathrm{pmay}}$ and $\simeq_{\mathrm{pmust}}$ respectively. These preorders, and their associated equivalences, were first defined by Wang and Larsen [YL92]. In the rest of this section, we will study them in detail.

Before doing so let us first point out a useful simplification: the Hoare and Smyth preorders on finite subsets of $[0, 1]$ (and more generally on *closed* subsets of $[0, 1]$) are determined by their maximum and minimum elements respectively.

Proposition 4.37 For $O_1,\ O_2 \in \mathcal{P}^+_{fin}(\mathcal{O}_{\mathrm{prob}})$ we have

(i) $O_1 \sqsubseteq_{\mathrm{Ho}} O_2$ if and only if $\max(O_1) \leq \max(O_2)$

(ii) $O_1 \sqsubseteq_{\mathrm{Sm}} O_2$ if and only if $\min(O_1) \leq \min(O_2)$.

Proof: Straightforward calculations. □

As in the non-probabilistic case [DNH84], we could also define a testing preorder combining the may-must-preorders; we will not study this combination here.

4.6.2 Testing probabilistic processes

Definition 4.38 *A* (probabilistic) process *is a tuple $\langle S, \Delta^\circ, Act_\tau, \rightarrow\rangle$, where $\langle S, Act_\tau, \rightarrow\rangle$ is a pLTS and Δ° is a distribution over S, called the* initial distribution *of the pLTS.*

A process is fully probabilistic *if each state has at most one outgoing transition.*

We now define the parallel composition of two processes.

Definition 4.39 *Let $P_1 = \langle S_1, \Delta_1^\circ, Act_\tau, \rightarrow_1\rangle$ and $P_2 = \langle S_2, \Delta_2^\circ, Act_\tau, \rightarrow_2\rangle$ be two processes, and A a set of visible actions. The* parallel composition $P_1|_A P_2$ *is $\langle S_1 \times S_2, \Delta_1^\circ \times \Delta_2^\circ, Act_\tau, \rightarrow\rangle$ where $\rightarrow$ is the least relation satisfying the following rules:*

$$\frac{s_1 \xrightarrow{\alpha}_1 \Delta}{(s_1, s_2) \xrightarrow{\alpha} \Delta \times \overline{s_2}} \quad \alpha \notin A \qquad\qquad \frac{s_2 \xrightarrow{\alpha}_2 \Delta}{(s_1, s_2) \xrightarrow{\alpha} \overline{s_1} \times \Delta} \quad \alpha \notin A$$

$$\frac{s_1 \xrightarrow{a}_1 \Delta_1,\ s_2 \xrightarrow{a}_2 \Delta_2}{(s_1, s_2) \xrightarrow{\tau} \Delta_1 \times \Delta_2} \quad a \in A$$

Parallel composition is the basis of testing: it models the interaction of the observer with the process being tested; and it models the observer himself — as a process. Let $\Omega := \{\omega_1, \omega_2, \cdots\}$ be a countable set of *success actions*, disjoint from Act_τ, define a *test* to be a process $\langle S, \Delta^\circ, Act_\tau \cup \Omega, \rightarrow\rangle$ with the constraint that $s \xrightarrow{\omega_i}$ and $s \xrightarrow{\omega_j}$ implies $i = j$. Let $\mathbb{T}$ be the class of all such tests, and write $\mathbb{T}_n$ for the subclass of $\mathbb{T}$ that uses only n success actions; we write $\mathbb{T}_\mathbb{N}$ for $\bigcup_{n\in\mathbb{N}} \mathbb{T}_n$.

To *apply* test T to process P we first form the composition $P \mid_{Act} T$ and then resolve all nondeterministic choices into probabilistic choices. Thus, we obtain a set of resolutions as in Definition 4.40 below. For each resolution, any particular success action ω_i will have some probability of occurring; and those probabilities, taken together, give us a single *success tuple* for the whole resolution, so that if o is the tuple then o_i is the recorded probability of ω_i's occurrence. The set of all those tuples, i.e. over all resolutions of $P \mid_{Act} T$, is then the complete outcome of applying test T to process P: as such, it will be a subset of $[0,1]^{\mathbb{N}^+}$.

Definition 4.40 *A* resolution *of a process $\langle S, \Delta^\circ, \rightarrow\rangle$ is a fully probabilistic process $\langle T, \Theta^\circ, \rightarrow\rangle$ such that there is a resolving function $f \in T \rightarrow S$ which satisfies the following conditions:*

1. *$f(\Theta^\circ) = \Delta^\circ$*
2. *if $t \xrightarrow{\omega} \Theta$ for some $\omega \in \Omega$ then $f(t) \xrightarrow{\omega} f(\Theta)$*
3. *if $t \xrightarrow{\alpha} \Theta$ for some $\alpha \notin \Omega$ then $f(t) \not\xrightarrow{\omega}$ for all $\omega \in \Omega$ and $f(t) \xrightarrow{\alpha} f(\Theta)$*
4. *if $t \not\rightarrow$ then $f(t) \not\rightarrow$*

where $f(\Theta)$ is the distribution defined by $f(\Theta)(s) = \sum_{f(t)=s} \Theta(t)$.

Given two tuples $o_1, o_2 \in [0,1]^\mathbb{N}$, we can compare them component-wise. Let S be a set, the set of functions $S \rightarrow [0,1]^\mathbb{N}$ forms a complete lattice when equipped with the partial order $\leq$ defined by: $f \leq g$ iff $f(s) \leq g(s)$ for all $s \in S$. Consider the function $F : (S \rightarrow [0,1]^\mathbb{N}) \rightarrow (S \rightarrow [0,1]^\mathbb{N})$ defined below; it extends to type $\mathcal{D}(S) \rightarrow [0,1]^\mathbb{N}$ via the convention $f(\Delta) := \mathrm{Exp}_\Delta(f)$.

$$F(f)(s) \ := \ \begin{cases} 1_i & \text{if } s \xrightarrow{\omega_i} \text{ for some } \omega_i \in \Omega \\ \vec{0} & \text{if } s \not\rightarrow \\ f(\Delta) & \text{otherwise} \end{cases} \tag{4.19}$$

where $\vec{0}$ is the everywhere-zero vector.

Lemma 4.41 *The function F defined above is continuous.*

Proof: Let $f_0 \leq f_1 \leq ...$ be an increasing chain in $S \to [0,1]^{\mathbb{N}}$. We need to show that

$$F(\bigsqcup_{n\geq 0} f_n) \;=\; \bigsqcup_{n\geq 0} F(f_n) \tag{4.20}$$

For any $s \in S$, we are in one of the following three cases:

1. $s \xrightarrow{\omega_i}$ for some $\omega_i \in \Omega$. We have

$$\begin{array}{rcll} F(\bigsqcup_{n\geq 0} f_n)(s) & = & 1_i & \text{by (4.19)} \\ & = & \bigsqcup_{n\geq 0} 1_i & \\ & = & \bigsqcup_{n\geq 0} F(f_n)(s) & \\ & = & (\bigsqcup_{n\geq 0} F(f_n))(s) & \end{array}$$

2. $s \not\longrightarrow$. Similar to last case. We have

$$F(\bigsqcup_{n\geq 0} f_n)(s) = \vec{0} = (\bigsqcup_{n\geq 0} F(f_n))(s).$$

3. Otherwise, $s \xrightarrow{\alpha} \Delta$ for some $\Delta \in \mathcal{D}(S)$ and $\alpha \in Act_\tau$. Then we infer that

$$\begin{array}{rcll} F(\bigsqcup_{n\geq 0} f_n)(s) & = & (\bigsqcup_{n\geq 0} f_n)(\Delta) & \text{by (4.19)} \\ & = & \sum_{s\in\lceil\Delta\rceil} \Delta(s) \cdot (\bigsqcup_{n\geq 0} f_n)(s) & \\ & = & \sum_{s\in\lceil\Delta\rceil} \Delta(s) \cdot (\bigsqcup_{n\geq 0} f_n(s)) & \\ & = & \sum_{s\in\lceil\Delta\rceil} \bigsqcup_{n\geq 0} \Delta(s) \cdot f_n(s) & \\ & = & \bigsqcup_{n\geq 0} \sum_{s\in\lceil\Delta\rceil} \Delta(s) \cdot f_n(s) & \text{by Lemma 1.13} \\ & = & \bigsqcup_{n\geq 0} f_n(\Delta) & \\ & = & \bigsqcup_{n\geq 0} F(f_n)(s) & \\ & = & (\bigsqcup_{n\geq 0} F(f_n))(s) & \end{array}$$

□

Because of Lemma 4.41 and Proposition 1.6, the function F has a least fixed point $\mathit{lfp}(F) = \sqcup_{n\in\mathbb{N}} F^n(\bot)$, where $\bot(s) = 0$ for all $s \in S$.

Definition 4.42 *Given a fully probabilistic process $\langle S, \Delta^\circ, \to\rangle$, we define a* results-gathering function *as follows:*

$$\mathbb{V}(\Delta^\circ) := \mathit{lfp}(F)(\Delta^\circ)$$

The results of applying Ω-test T to process P is given by:

$$\mathcal{A}\mathit{pply}(T,P) := \{\mathbb{V}(\Delta^\circ) \mid \Delta^\circ \text{ is the initial distribution of a resolution of } P \mid_{Act} T\}$$

We note that the result set $\mathcal{A}\mathit{pply}(T,P)$ is convex.

Lemma 4.43 *For any test T and process P, if $o_1, o_2 \in \mathcal{A}\mathit{pply}(T,P)$, then their weighted average $o_1 {}_p\!\oplus o_2$ is also in $\mathcal{A}\mathit{pply}(T,P)$ for any $p \in [0,1]$.*

Proof: Let R_1, R_2 be the resolutions of $P \mid_{Act} T$ that gave rise to o_1, o_2. Form R as their disjoint union, except initially we define $\Delta^\circ := \Delta_1^\circ {}_p\!\oplus \Delta_2^\circ$, where $\Delta^\circ, \Delta_1^\circ, \Delta_2^\circ$ are the initial distributions of R, R_1, R_2 respectively. The new resolution R generates the interpolated tuple $o_1 {}_p\!\oplus o_2$ as required. □

Definition 4.44 *Let P and Q be two probabilistic processes.*

$$\begin{array}{lll} P \sqsubseteq^n_{pmay} Q & \textit{iff} & \forall T \in \mathbb{T}_n : \mathcal{A}\mathit{pply}(T,P) \sqsubseteq_{\mathrm{Ho}} \mathcal{A}\mathit{pply}(T,Q) \\ P \sqsubseteq^n_{pmust} Q & \textit{iff} & \forall T \in \mathbb{T}_n : \mathcal{A}\mathit{pply}(T,P) \sqsubseteq_{\mathrm{Sm}} \mathcal{A}\mathit{pply}(T,Q) \end{array}$$

4.7 Reward testing

In this section we introduce a new way of testing based on giving rewards to success actions.

We begin with some notation. A *reward tuple* is an n-tuple $h \in \mathbb{R}^n$ of real numbers. Given two tuples $h, o \in \mathbb{R}^n$, we write $h \cdot o$ for the dot-product of them. Given a set of tuples $O \in [0,1]^n$ and a reward tuple $h \in \mathbb{R}^n$, we write $h \cdot O$ for the set of rewards $\{h \cdot o \mid o \in O\}$.

4.7.1 A geometric property

Definition 4.45 *A subset O of the n-dimensional Euclidean space is* p-closed *(for* probabilistically *closed) iff*

- *It is* convex*, that is if $o_1, o_2 \in O$ and $p \in [0,1]$ then the weighted average $o_1 \,{}_p\!\oplus o_2$ is also in O, and*
- *It is* Cauchy closed*, that is it contains all its limit points in the usual Euclidean metric, and it is* bounded. [2]

P-closure allows us to appeal to the *Separating Hyperplane Lemma* from discrete geometry [Mat02, Theorem 1.2.4 paraphrased]:

Lemma 4.46 *Let A and B be two convex- and Cauchy-closed subsets of Euclidean n-space; assume that they are disjoint and that at least one of them is bounded. Then there is a hyperplane that strictly separates them.*

Here a *hyperplane* is a set of the form $\{w \in \mathbb{R}^n \mid h \cdot w = c\}$ for certain $h \in \mathbb{R}^n$ (the *normal* of the hyperplane) and $c \in \mathbb{R}$, and such a hyperplane *strictly separates* A and B if for all $a \in A$ and $b \in B$ we have $h \cdot a < c < h \cdot b$ or $h \cdot a > c > h \cdot b$.

Theorem 4.47 *Let A, B be subsets of $[0,1]^n$; then we have*

$$\begin{array}{lll} A \sqsubseteq_{\mathrm{Ho}} B & \textit{iff} \quad \forall h \in [0,1]^n : \bigsqcup h \cdot A \leq \bigsqcup h \cdot B & \textit{if } B \textit{ is p-closed, and} \\ A \sqsubseteq_{\mathrm{Sm}} B & \textit{iff} \quad \forall h \in [0,1]^n : \bigsqcap h \cdot A \leq \bigsqcap h \cdot B & \textit{if } A \textit{ is p-closed.} \end{array}$$

Proof: We consider first the *only-if* -direction for the Smyth case:

$$\begin{array}{lll} & A \sqsubseteq_{\mathrm{Sm}} B & \\ \Leftrightarrow & \forall b \in B : \exists a \in A : a \leq b & [\text{Definition of } \sqsubseteq_{\mathrm{Ho}}] \\ \Rightarrow & \forall h \in [0,1]^n : \forall b \in B : \exists a \in A : h \cdot a \leq h \cdot b & [h \geq 0] \\ \Rightarrow & \forall h \in [0,1]^n : \forall b \in B : \bigsqcap h \cdot A \leq h \cdot b & [\bigsqcap h \cdot A \leq h \cdot a] \\ \Rightarrow & \forall h \in [0,1]^n : \bigsqcap h \cdot A \leq \bigsqcap h \cdot B & [\text{Definition of infimum}] \end{array}$$

For the *if*-direction we use separating hyperplanes, proving the contrapositive:

$$\begin{array}{lll} & A \not\sqsubseteq_{\mathrm{Sm}} B & \\ \Leftrightarrow & \forall a \in A : \neg(a \leq b) & [\text{Definition of } \sqsubseteq_{\mathrm{Sm}}\text{; for some } b \in B] \\ \Leftrightarrow & A \cap B' = \emptyset & [\text{define } B' := \{b' \in \mathbb{R}^n \mid b' \leq b\}] \\ \Leftrightarrow & \exists h \in \mathbb{R}^n, c \in \mathbb{R} : & [\text{Lemma 4.46; } A \text{ is p-closed; } B' \text{ is convex and Cauchy-closed}] \\ & \quad \forall a \in A, b' \in B' : & \\ & \quad\quad h \cdot b' < c < h \cdot a & \end{array}$$

where without loss of generality the inequality can be in the direction shown, else we simply multiply h, c by -1.

We now argue that h is non-negative, whence by scaling of h, c we obtain without loss of generality that $h \in [0,1]^n$. Assume for a contradiction that $h_i < 0$. Choose scalar $d \geq 0$ large enough so that the

[2] Cauchy closure and boundedness together amounts to *compactness*.

point $b' := (b_1, \cdots, b_i - d, \cdots, b_n)$ falsifies $h \cdot b' < c$; since b' is still in B', however, that contradicts the separation. Thus we continue

$$\begin{array}{lll}
 & \exists h \in [0,1]^n, c \in \mathbb{R} : & \\
\Leftrightarrow & \quad \forall a \in A, b' \in B' : & \text{[above comments concerning } d\text{]} \\
 & \quad\quad h \cdot b' < c < h \cdot a & \\
\Leftrightarrow & \exists h \in [0,1]^n, c \in \mathbb{R} : \forall a \in A : h \cdot b < c < h \cdot a & \text{[set } b' \text{ to } b\text{; note } b \in B'\text{]} \\
\Rightarrow & \exists h \in [0,1]^n, c \in \mathbb{R} : h \cdot b < c \leq \bigsqcap h \cdot A & \text{[property of infimum]} \\
\Rightarrow & \exists h \in [0,1]^n, c \in \mathbb{R} : \bigsqcap h \cdot B < c \leq \bigsqcap h \cdot A & \text{[} b \in B\text{, hence } \bigsqcap h \cdot B \leq h \cdot b\text{]} \\
\Rightarrow & \neg(\forall h \in [0,1]^n : \bigsqcap h \cdot A \leq \bigsqcap h \cdot B) &
\end{array}$$

The proof for the Hoare-case is analogous. □

4.7.2 Reward testing

Definition 4.48 *Let P and Q be two probabilistic processes. We define two reward testing preorders.*

$$\begin{array}{lll}
P \sqsubseteq^n_{rmay} Q & \textit{iff} & \forall T \in \mathbb{T}_n, \forall h \in [0,1]^n : \bigsqcup h \cdot \mathcal{A}pply(T,P) \leq \bigsqcup h \cdot \mathcal{A}pply(T,Q) \\
P \sqsubseteq^n_{rmust} Q & \textit{iff} & \forall T \in \mathbb{T}_n, \forall h \in [0,1]^n : \bigsqcap h \cdot \mathcal{A}pply(T,P) \leq \bigsqcap h \cdot \mathcal{A}pply(T,Q)
\end{array}$$

Definition 4.49 *Given a fully probabilistic process with state set S, we define the function $\mathbb{V}^\delta_k : S \to [0,1]^m$ to calculate the vector of success probabilities for each state with respect to the discount factor $\delta \in (0,1]$; it extends to type $\mathcal{D}(S) \to [0,1]^m$ via the convention $\mathbb{V}^\delta_k(\Delta) := Exp_\Delta(\mathbb{V}^\delta_k)$.*

- $\mathbb{V}^\delta_0(s) \;:=\; \begin{cases} 1_i & \textit{if } s \xrightarrow{\omega_i} \\ \vec{0} & \textit{otherwise} \end{cases}$
- $\mathbb{V}^\delta_{(k+1)}(s) \;:=\; \begin{cases} 1_i & \textit{if } s \xrightarrow{\omega_i} \textit{ for some } \omega_i \in \Omega \\ \vec{0} & \textit{if } s \not\longrightarrow \\ \delta \cdot \mathbb{V}_k(\Delta) & \textit{otherwise} \end{cases}$

where 1_i is the vector that is 1 at position i and 0 elsewhere, while $\vec{0}$ is the zero vector.

Given a reward tuple $h \in [0,1]^m$, we define a scalar outcome $\mathbb{V}^{\delta,h}_k : S \to [0,1]$ by taking the inner product of two vectors:

$$\mathbb{V}^{\delta,h}_k(s) \;:=\; h \cdot \mathbb{V}^\delta_k(s)$$

By construction, for every state s we have

- $\mathbb{V}^{\delta,h}_k(s) \leq 1$
- $\mathbb{V}^{\delta,h}_k(s) \leq \mathbb{V}^{\delta,h}_{(k+1)}(s)$

Therefore, we can define $\mathbb{V}^{\delta,h} : S \to [0,1]$ by letting $\mathbb{V}^{\delta,h}(s) := \lim_{k\to\infty} \mathbb{V}^{\delta,h}_k(s)$.

Consider the set of all functions from S to $[0,1]$, denoted by $\mathcal{F}_S$, and the distance function d over $\mathcal{F}_S$ defined by $d(f,g) = max_{s\in S}|f(s) - g(s)|$. We can check that $(\mathcal{F}_S, d)$ constitutes a complete metric space (cf. Example 1.19). The functional $F^{\delta,h} : \mathcal{F}_S \to \mathcal{F}_S$ defined by

$$F^{\delta,h}(f)(s) = \begin{cases} h_i & \text{if } s \xrightarrow{\omega_i} \\ 0 & \text{if } s \not\longrightarrow \\ \delta \cdot f(\Delta) & \text{otherwise} \end{cases}$$

where $f(\Delta) = \mathrm{Exp}_\Delta(f)$ and $\delta \in (0,1)$, is a contraction mapping with constant δ. It follows from Theorem 1.21 that $F^{\delta,h}$ has a unique fixed point, which is the limit of the sequence $(F^k(\mathbb{V}^{\delta,h}_0))^\infty_{k=0}$, that is $\mathbb{V}^{\delta,h}$.

4.7.3 Maximum rewards

Definition 4.50 *Given a probabilistic process with state set S and a reward tuple $h \in [0,1]^m$, we define the function $\mathbb{V}max_k^{\delta,h} : S \to [0,1]^m$ to calculate the maximum reward for each state with respect to the discount factor $\delta \in (0,1]$.*

- $\mathbb{V}max_0^{\delta,h}(s) \;:=\; \begin{cases} h_i & \text{if } s \xrightarrow{\omega_i} \\ 0 & \text{otherwise} \end{cases}$
- $\mathbb{V}max_{(k+1)}^{\delta,h}(s) \;:=\; \begin{cases} h_i & \text{if } s \xrightarrow{\omega_i} \\ 0 & \text{if } s \not\longrightarrow \\ \delta \cdot max\{\mathbb{V}max_k^{\delta,h}(\Delta) \mid s \xrightarrow{\alpha} \Delta\} & \text{otherwise} \end{cases}$

where $\mathbb{V}max_k^{\delta,h}(\Delta) = Exp_\Delta(\mathbb{V}max_k^{\delta,h})$ and h_i is the i-th component of the tuple h.

By construction, for every state s we have

- $\mathbb{V}max_k^{\delta,h}(s) \leq 1$
- $\mathbb{V}max_k^{\delta,h}(s) \leq \mathbb{V}max_{(k+1)}^{\delta,h}(s)$

Therefore, we can define $\mathbb{V}max^{\delta,h} : S \to [0,1]$ by letting $\mathbb{V}max^{\delta,h}(s) := \lim_{k\to\infty} \mathbb{V}max_k^{\delta,h}(s)$.

Similar to $\mathbb{V}^{\delta,h}$, the function $\mathbb{V}max^{\delta,h}$ is the unique fixed point of the functional $Fmax^{\delta,h} : \mathcal{F}_S \to \mathcal{F}_S$ defined by

$$Fmax^{\delta,h}(f)(s) = \begin{cases} h_i & \text{if } s \xrightarrow{\omega_i} \\ 0 & \text{if } s \not\longrightarrow \\ \delta \cdot max\{f(\Delta) \mid s \xrightarrow{\alpha} \Delta\} & \text{otherwise.} \end{cases}$$

Lemma 4.51 *1.* $\mathbb{V}max^{\delta,h}(\Delta) = \sum_{s\in S} \Delta(s) \cdot \mathbb{V}max^{\delta,h}(s)$

2. $\mathbb{V}^{\delta,h}(\Theta) = \sum_{t\in T} \Theta(t) \cdot \mathbb{V}^{\delta,h}(t)$

□

Lemma 4.52 *Suppose $\delta \in (0,1]$ and $h \in [0,1]^m$. If $(T, \Theta^\circ, \to)$ is a resolution of $(S, \Delta^\circ, \to)$, then $\mathbb{V}max^{\delta,h}(\Delta^\circ) \geq \mathbb{V}^{\delta,h}(\Theta^\circ)$.*

Proof: Let $f : T \to S$ be the resolving function associated with the resolution $(T, \Theta^\circ, \to)$, we show by induction on k that

$$\mathbb{V}max_k^{\delta,h}(f(t)) \geq \mathbb{V}_k^{\delta,h}(t) \text{ for any } t \in T \tag{4.21}$$

The case $k = 0$ is trivial. For the inductive step, the interesting case is when $t \xrightarrow{\tau} \Theta$. Then $f(t) \not\xrightarrow{\omega}$ for all $\omega \in \Omega$ and $f(t) \xrightarrow{\tau} f(\Theta)$. We can infer that

$$\begin{array}{rcll} \mathbb{V}max_{(k+1)}^{\delta,h}(f(t)) & = & \delta \cdot max\{\mathbb{V}max_k^{\delta,h}(\Delta) | f(t) \xrightarrow{\tau} \Delta\} & \\ & \geq & \delta \cdot \mathbb{V}max_k^{\delta,h}(f(\Theta)) & \\ & = & \delta \cdot \sum_{s\in S} f(\Theta)(s) \cdot \mathbb{V}max_k^{\delta,h}(s) & \\ & = & \delta \cdot \sum_{t\in T} \Theta(t) \cdot \mathbb{V}max_k^{\delta,h}(f(t)) & \\ & \geq & \delta \cdot \sum_{t\in T} \Theta(t) \cdot \mathbb{V}_k^{\delta,h}(t) & \text{by induction} \\ & = & \delta \cdot \mathbb{V}_k^{\delta,h}(\Theta) & \\ & = & \mathbb{V}_{(k+1)}^{\delta,h}(t) & \end{array}$$

It follows from (4.21) that

$$\mathbb{V}max^{\delta,h}(f(t)) \geq \mathbb{V}^{\delta,h}(t) \text{ for any } t \in T \tag{4.22}$$

Therefore, we have that

$$\begin{array}{rcll}
\mathbb{V}max^{\delta,h}(\Delta^\circ) & = & \mathbb{V}max^{\delta,h}(f(\Theta^\circ)) & \\
& = & \sum_{s\in S} f(\Theta^\circ)(s)\cdot \mathbb{V}max^{\delta,h}(s) & \text{by Lemma 4.51 (2)} \\
& = & \sum_{t\in T} \Theta^\circ(t)\cdot \mathbb{V}max^{\delta,h}(f(t)) & \\
& \geq & \sum_{t\in T} \Theta^\circ(t)\cdot \mathbb{V}^{\delta,h}(t) & \text{by (4.22)} \\
& = & \mathbb{V}^{\delta,h}(\Theta^\circ) & \text{by Lemma 4.51 (3)}
\end{array}$$

□

We say a resolution of a process is *static* if its associated resolving function is injective.

Lemma 4.53 *Suppose $\delta \in (0,1)$ and $h \in [0,1]^m$. Given a process $(S,\Delta^\circ,\rightarrow)$, there exists a static resolution $(T,\Theta^\circ,\rightarrow)$ such that $\mathbb{V}max^{\delta,h}(\Delta^\circ) = \mathbb{V}^{\delta,h}(\Theta^\circ)$.*

Proof: Let $(T,\Theta^\circ,\rightarrow)$ be a resolution with an injective resolving function f such that if $t \xrightarrow{\tau} \Theta$ then $\mathbb{V}max^{\delta,h}(f(\Theta)) = max\{\mathbb{V}max^{\delta,h}(\Delta) \mid f(t) \xrightarrow{\tau} \Delta\}$. The assumption of finite branchingness ensures the existence of such resolving function f.

Let $g : T \rightarrow [0,1]$ be the function defined by $g(t) = \mathbb{V}max^{\delta,h}(f(t))$. Below we show that g is a fixed point of $F^{\delta,h}$. We only consider the non-trivial case that $t \xrightarrow{\tau} \Theta$. By the definition of f, we have $f(t) \not\xrightarrow{\omega}$ for all $\omega \in \Omega$, $f(t) \xrightarrow{\tau} f(\Theta)$ with $\mathbb{V}max^{\delta,h}(f(\Theta)) = max\{\mathbb{V}max^{\delta,h}(\Delta) \mid f(t) \xrightarrow{\tau} \Delta\}$. Therefore,

$$\begin{array}{rcl}
F^{\delta,h}(g)(t) & = & \delta\cdot g(\Theta) \\
& = & \delta\cdot \sum_{t\in T}\Theta(t)\cdot g(t) \\
& = & \delta\cdot \sum_{t\in T}\Theta(t)\cdot \mathbb{V}max^{\delta,h}(f(t)) \\
& = & \delta\cdot \sum_{s\in S} f(\Theta)(s)\cdot \mathbb{V}max^{\delta,h}(s) \\
& = & \delta\cdot \mathbb{V}max^{\delta,h}(f(\Theta)) \\
& = & \delta\cdot max\{\mathbb{V}max^{\delta,h}(\Delta) | f(t) \xrightarrow{\tau} \Delta\} \\
& = & \mathbb{V}max^{\delta,h}(f(t)) \\
& = & g(t)
\end{array}$$

Since $F^{\delta,h}$ has a unique fixed point $\mathbb{V}^{\delta,h}$, we derive that g coincides with $\mathbb{V}^{\delta,h}$, i.e. $\mathbb{V}^{\delta,h}(t) = g(t) = \mathbb{V}max^{\delta,h}(f(t))$ for all $t \in T$, from which we can obtain the required result $\mathbb{V}^{\delta,h}(\Theta^\circ) = \mathbb{V}max^{\delta,h}(\Delta^\circ)$. □

Lemma 4.54 *Let $\{\delta_n\}_{n\geq 1}$ be a nondecreasing sequence of discount factors converging to 1.*

- $\mathbb{V}max^{1,h} = \lim_{n\rightarrow\infty} \mathbb{V}max^{\delta_n,h}$
- $\mathbb{V}^{1,h} = \lim_{n\rightarrow\infty} \mathbb{V}^{\delta_n,h}$

□

Theorem 4.55 *Suppose $h \in [0,1]^m$. Given a process $(S,\Delta^\circ,\rightarrow)$, there exists a static resolution $(T,\Theta^\circ,\rightarrow)$ such that $\mathbb{V}max^{1,h}(\Delta^\circ) = \mathbb{V}^{1,h}(\Theta^\circ)$.*

Proof: By Lemma 4.53, for every discount factor $d \in (0,1)$ there exists a static resolution which achieves the maximum reward. Since there are finitely many states in S, there are finitely many static resolutions. There must exist a static resolution that achieves the maximum rewards for infinitely many discount factors. In other words, for every nondecreasing sequence $\{\delta_n\}_{n\geq 1}$ converging to 1, there exists a subsequence $\{\delta_{n_k}\}_{k\geq 1}$ and a static resolution $(T,\longrightarrow,\Theta^\circ)$ with resolving function f_0 such that $\mathbb{V}^{\delta_{n_k},h}(t) = \mathbb{V}max^{\delta_{n_k},h}(f_0(t))$ for all $t \in T$ and $k = 1,2,....$ By Lemma 4.54, we have that, for every $t \in T$,

$$\begin{array}{rcl}
\mathbb{V}^{1,h}(t) & = & \lim_{k\rightarrow\infty} \mathbb{V}^{\delta_{n_k},h}(t) \\
& = & \lim_{k\rightarrow\infty} \mathbb{V}max^{\delta_{n_k},h}(f_0(t)) \\
& = & \mathbb{V}max^{1,h}(f_0(t))
\end{array}$$

It follows that $\mathbb{V}^{1,h}(\Theta^\circ) = \mathbb{V}max^{1,h}(\Delta^\circ)$. □

4.7.4 Minimum rewards

We now consider probabilistic processes which are finitely branching and can have countably many states. The dual of $\mathbb{V}max^{\delta,h}$ is $\mathbb{V}min^{\delta,h}$ by replacing max with min in Definition 4.50. In fact, we do not need discount factors any more. So we just write $\mathbb{V}min^h$ for $\mathbb{V}min^{1,h}$, $\mathbb{V}^h$ for $\mathbb{V}^{1,h}$, and F^h for $F^{1,h}$.

Lemma 4.56 *For any $h \in [0,1]^m$, if $(T, \Theta^\circ), \rightarrow$ is a resolution of $(S, \Delta^\circ, \rightarrow)$, then it holds that $\mathbb{V}min^h(\Delta^\circ) \leq \mathbb{V}^h(\Theta^\circ)$.*

Proof: Analogous to the proof of Lemma 4.52. □

Theorem 4.57 *Suppose $h \in [0,1]^m$. Given a process $(S, \longrightarrow, \Delta^\circ)$. There exists a static resolution $(T, \Theta^\circ, \rightarrow)$ such that $\mathbb{V}min^h(\Delta^\circ) = \mathbb{V}^h(\Theta^\circ)$.*

Proof: Similar to the proof of Lemma 4.53. Let $(T, \Theta^\circ, \rightarrow)$ be a resolution with an injective resolving function f such that if $t \xrightarrow{\tau} \Theta$ then $\mathbb{V}min^h(f(\Theta)) = min\{\mathbb{V}min^h(\Delta) \mid f(t) \xrightarrow{\tau} \Delta\}$.

Let $g : T \rightarrow [0,1]$ be the function defined by $g(t) = \mathbb{V}min^{\delta,h}(f(t))$. As in the proof of Lemma 4.53, we show that g is a fixed point of F^h. It is also easy to prove that $\mathbb{V}^h$ is the least fixed point of F^h. Therefore, $\mathbb{V}^h(t) \leq g(t) = \mathbb{V}min^h(f(t))$ for all $t \in T$, from which we can obtain $\mathbb{V}^h(\Theta^\circ) \leq \mathbb{V}min^h(\Delta^\circ)$. Using Lemma 4.56, we derive that $\mathbb{V}min^h(\Delta^\circ) = \mathbb{V}^h(\Theta^\circ)$. □

From Lemmas 4.52, 4.56 and Theorems 4.55 and 4.57, we obtain the following corollary.

Corollary 4.58 *Let $h \in [0,1]^m$ and $M = (S, \Delta^\circ, \rightarrow)$ be a finitely branching probabilistic process.*

1. *If S is finite, then*

$$\mathbb{V}max^h = max\{\mathbb{V}^h(\Theta^\circ) \mid \Theta^\circ \text{ is the initial distribution of a resolution of } M\}$$

2. *If S is countable, then*

$$\mathbb{V}min^h = min\{\mathbb{V}^h(\Theta^\circ) \mid \Theta^\circ \text{ is the initial distribution of a resolution of } M\}$$

So the two ways of calculating maximum/minimum rewards coincide: one is to do the calculation on a probabilistic process directly ($\mathbb{V}max^h$ and $\mathbb{V}min^h$), the other is to calculate the reward of each resolution of the process ($\mathbb{V}^h$) and then take the maximum/minimum rewards.

As a remark, in the special case of having only one success action, as in [DvGH$^+$07a, DvGH$^+$07b], $\mathbb{V}max^h$ and $\mathbb{V}min^h$ give the maximum and minimum success probabilities, respectively, by taking $h = 1$.

4.7.5 Vector testing versus scalar testing

Definition 4.59 *The results of applying Ω-test T to process P is given by:*

$$\mathcal{A}pply_f(T, P) \;:=\; \updownarrow\{\mathbb{V}(\Delta^\circ) \mid \Delta^\circ \text{ is the initial distribution of a static resolution of } P \mid_{Act} T\}$$

A process is *finitary* if it is finitely branching and has finitely many states.

Lemma 4.60 *For any test T and process P, we have $\mathcal{A}pply_f(T, P) \subseteq \mathcal{A}pply(T, P)$. If P and T are finitary, then $\mathcal{A}pply_f(T, P)$ is p-closed.*

Proof: The first statement is trivial, since static resolutions are still resolutions and Lemma 4.43 maintains that $\mathcal{A}pply(T, P)$ is convex. If P and T are finitary, their composition $P \mid_{Act} T$ is finitary too. The set $\mathcal{A}pply_f(T, P)$ is the convex closure of a finite number of points, so it is clearly Cauchy closed. □

Lemma 4.61 *Let $h \in [0,1]^m$ be a reward tuple, $T \in \mathbb{T}_m$ and P are finitary test and process, respectively.*

$$\begin{array}{rcl} \bigsqcup h \cdot \mathcal{A}pply_f(T,P) & = & \bigsqcup h \cdot \mathcal{A}pply(T,P) \\ \bigsqcap h \cdot \mathcal{A}pply_f(T,P) & = & \bigsqcap h \cdot \mathcal{A}pply(T,P) \end{array}$$

Proof: Lemma 4.52 and Theorem 4.55 imply that

$$\bigsqcup h \cdot \mathcal{A}pply_f(T,P) \;=\; \mathbb{V}max^{1,h}(T,P) \;=\; \bigsqcup h \cdot \mathcal{A}pply(T,P) \tag{4.23}$$

Similarly, Lemma 4.56 and Theorem 4.57 imply that

$$\bigsqcap h \cdot \mathcal{A}pply_f(T,P) \;=\; \mathbb{V}max^{1,h}(T,P) \;=\; \bigsqcap h \cdot \mathcal{A}pply(T,P) \tag{4.24}$$

□

Lemma 4.62 *For any $n \in \mathbb{N}$ and finitary processes P, Q we have*

$$\begin{array}{rcl} P \sqsubseteq^n_{pmay} Q & \textit{iff} & P \sqsubseteq^n_{rmay} Q \\ P \sqsubseteq^n_{pmust} Q & \textit{iff} & P \sqsubseteq^n_{rmust} Q \end{array}$$

Proof: For the *only-if*-direction, we apply Theorem 4.47; this direction does not require p-closure.
For the *if*-direction, we prove the must-case in the contrapositive; the may-case is similar.

$$\begin{array}{lll} & P \not\sqsubseteq^n_{\mathrm{pmust}} Q & \\ \Leftrightarrow & \mathcal{A}pply(T,P) \not\sqsubseteq_{\mathrm{Sm}} \mathcal{A}pply(T,Q) & [\text{for some } T \in \mathbb{T}_n] \\ \Rightarrow & \mathcal{A}pply_f(T,P) \not\sqsubseteq_{\mathrm{Sm}} \mathcal{A}pply(T,Q) & [\text{Lemma 4.60}] \\ \Leftrightarrow & \bigsqcap h \cdot \mathcal{A}pply_f(T,P) \not\sqsubseteq_{\mathrm{Sm}} \bigsqcap h \cdot \mathcal{A}pply(T,Q) & [\text{Lemma 4.60; Theorem 4.47}] \\ \Leftrightarrow & \bigsqcap h \cdot \mathcal{A}pply(T,P) \not\sqsubseteq_{\mathrm{Sm}} \bigsqcap h \cdot \mathcal{A}pply(T,Q) & [\text{for all } h \in [0,1]^n; \text{ Lemma 4.61}] \\ \Rightarrow & P \not\sqsubseteq^n_{\mathrm{rmust}} Q & \end{array}$$

□

We now show that for finitary processes scalar testing is equally powerful as finite-dimensional reward testing. In doing so, we assume that tests are ω-terminal in the sense that they halt after execution of any success action. [3]

Theorem 4.63 *For any $n \in \mathbb{N}$ and finitary processes P, Q we have*

$$\begin{array}{rcl} P \sqsubseteq^n_{rmay} Q & \textit{iff} & P \sqsubseteq^1_{rmay} Q \\ P \sqsubseteq^n_{rmust} Q & \textit{iff} & P \sqsubseteq^1_{rmust} Q \end{array}$$

Proof: The *only-if*-direction is trivial in both cases. For *if* we prove the must-case in the contrapositive; the may-case is similar.

Suppose thus that $P \not\sqsubseteq^n_{\mathrm{rmust}} Q$, then P, Q are distinguished by some test $T \in \mathbb{T}_n$ and reward $h \in [0,1]^n$, so that

$$\bigsqcap h \cdot \mathcal{A}pply(T,P) \;\not\leq\; \bigsqcap h \cdot \mathcal{A}pply(T,Q). \tag{4.25}$$

Without loss of generality, we assume that the success actions are $\omega_1, \cdots, \omega_n$ and construct a process U such that

- The state space is $\{u_0, \cdots, u_n\}$ together with a deadlock state u,
- The actions are $\omega_1, \cdots \omega_n$ and ω, all external,
- The initial distribution is $\overline{u_0}$ and
- The transitions are $u_0 \xrightarrow{\omega_i} \overline{u_i}_{\,h_i} \oplus \overline{u}$ and $u_i \xrightarrow{\omega} \overline{u}$, for $1 \leq i \leq n$.

[3] This assumption is justified in Section 4.7.6.

We now consider the test $T|_\Omega U$ with ω as its only success action. In $T|_\Omega U$ an occurrence of ω_i is with probability h_i followed immediately by an occurrence of ω (and with probability $1-h_i$ by deadlock). For any process P, the overall probability of ω's occurrence, in any resolution of $P\,|_{Act}\,(T|_\Omega U)$, is therefore the h-weighted reward $h\cdot o$ for the tuple o in the corresponding resolution of $P\,|_{Act}\,T$.

Thus from (4.25) we have that P, Q can be distinguished using the scalar test $T|_\Omega U$ with its single success action ω; that is, we achieve $P \not\sqsubseteq^1_{\text{rmust}} Q$ as required. □

Combining Lemma 4.62 and Theorem 4.63 yields that scalar testing is as powerful as finite-dimensional vector-based testing.

Theorem 4.64 *For any $n \in \mathbb{N}$ and finitary processes P, Q we have*

$$P \sqsubseteq^n_{pmay} Q \quad \textit{iff} \quad P \sqsubseteq^1_{pmay} Q$$
$$P \sqsubseteq^n_{pmust} Q \quad \textit{iff} \quad P \sqsubseteq^1_{pmust} Q$$

□

4.7.6 One success never leads to another

Here we substantiate the claim made in Section 4.7.5 (Footnote 3) that without loss of generality we can assume that our tests halt after engaging in any success action. The reward-testing construction requires this because the test U used in Theorem 4.63 implementing a particular reward-tuple h effectively causes the composite test $T|_\Omega U$ to halt after the reward is applied — hence for correctness of the construction we required that the original T must have halted anyway.

Below we show that the may- and must testing preorders do not change upon restricting the class of available tests to those that cannot perform multiple success actions in a single run. A second reduction to tests that actually halt after performing any success action is trivial: just change any transition of the form $s \xrightarrow{\omega_i} \Delta$ into a transition $s \xrightarrow{\omega_i} \overline{0}$ leading to a deadlocked state 0.

Suppose our original test T has n success actions $\omega_1, \cdots, \omega_n$. By running it in parallel with another test V (below) we will convert it to a new test with the required property and corresponding success actions $\omega'_1, \cdots, \omega'_n$, and with success n-tuples that are exactly $1/n$ times the success tuples of T; since testing is insensitive to scaling of the tuples, that will give us our result. Note that the $1/n$ factor is natural given that we are making our n success actions mutually exclusive: it ensures the tuple's elements' total does not exceed one.

We construct the test $V := \langle S, \Delta^\circ, \rightarrow\rangle$ as follows:

- The state space is $S := \mathcal{P}(\{\omega_1, \ldots, \omega_n\}) \cup \{0, \ldots, n\}$, where the powerset-states record which success actions have already occurred, the scalar states $1, \ldots, n$ are "about-to-terminate" states, and 0 is a deadlock state;
- The actions are $\omega_1, \ldots, \omega_n$ and $\omega'_1, \ldots, \omega'_n$, all external; and
- The initial distribution Δ° is the point distribution $\overline{\emptyset}$.

The transitions of V are of three kinds:

- "Terminating" transitions take state n with probability one via action ω'_n to the deadlocked state 0, i.e. $n \xrightarrow{\omega'_n} \overline{0}$;
- "Do-nothing" transitions, from state $s \in \mathcal{P}(\{\omega_1, \cdots, \omega_n\})$, lead with probability one via action $\omega_i \in s$ back to s, implementing that second and subsequent occurrences of any success action in T can be ignored, i.e. $s \xrightarrow{\omega_i} \overline{s}$ for $\omega_i \in s$; and
- "Success" transitions, from state $s \in \mathcal{P}(\{\omega_1, \cdots, \omega_n\})$ lead via action $\omega_i \notin s$ with probability $\frac{1}{n-\#s}$ to state i, whence the subsequent terminating transition will emit ω'_i; the remaining probability at s leads to state $s \cup \{\omega_i\}$, recording silently that ω_i has now been taken care of. That is, $s \xrightarrow{\omega_i} \overline{i} \;\frac{1}{n-\#s}\!\oplus\; \overline{s \cup \{\omega_i\}}$ for $\omega_i \notin s$.

When the original test T has finitely many states and transitions, so does the composite test $T|_\Omega V$.

Chapter 5

Testing Finite Probabilistic Processes

5.1 Introduction

A satisfactory semantic theory for processes which encompass both nondeterministic and probabilistic behaviour has been a long-standing research problem [HJ90, YL92, Low93, JHSY94, SL94, Sei95, Seg95, JY95, MMSS96, Seg96, HSM97, KN98, Mis00, BS01, JY02, LSV03, CCR+03a, TKP05, DP07]. In 1992 Wang & Larsen posed the problems of finding complete axiomatisations and alternative characterisations for a natural generalisation of the standard testing preorders [DNH84] to such processes [YL92]. Here we solve both problems, at least for finite processes, by providing a detailed account of both may- and must testing preorders for a finite version of the process calculus CSP extended with probabilistic choice. For each preorder we provide three independent characterisations, using (i) co-inductive simulation relations, (ii) a modal logic and (iii) sets of inequations.

Testing processes: Our starting point is the finite process calculus pCSP [DvGH+07a] obtained by adding a probabilistic choice operator to finite CSP; like others who have done the same, we now have *three* choice operators, external $P \,\Box\, Q$, internal $P \sqcap Q$ and the newly added probabilistic choice $P \,{}_p\!\oplus\, Q$. So a semantic theory for pCSP will have to provide a coherent account of the precise relationships between these operators.

The object of this chapter is to give alternative characterisations of these testing preorders. This problem was addressed previously by Segala in [Seg96], but using testing preorders ($\widehat{\sqsubseteq}^{\Omega}_{\text{pmay}}$ and $\widehat{\sqsubseteq}^{\Omega}_{\text{pmust}}$) that differ in two ways from the ones in [DNH84, Hen88, YL92, DvGH+07a] and the present chatper. First of all, in [Seg96] the success of a test is achieved by the *actual execution* of a predefined *success action*, rather than the reaching of a *success state*. We call this an *action-based* approach, as opposed to the *state-based* approach used in this chatper. Secondly, [Seg96] employs a countable number of success actions instead of a single one; we call this *vector-based*, as opposed to *scalar*, testing. Segala's results in [Seg96] depend crucially on this form of testing. To achieve our current results, we need Segala's preorders as a stepping stone. We relate them to ours by considering intermediate preorders $\widehat{\sqsubseteq}_{\text{pmay}}$ and $\widehat{\sqsubseteq}_{\text{pmust}}$ that arise from action-based but scalar testing, and use Theorem 4.64 from Chapter 4 saying that for finite processes the preorders $\widehat{\sqsubseteq}^{\Omega}_{\text{pmay}}$ and $\widehat{\sqsubseteq}^{\Omega}_{\text{pmust}}$ coincide with $\widehat{\sqsubseteq}_{\text{pmay}}$ and $\widehat{\sqsubseteq}_{\text{pmust}}$. Here we show that on pCSP the preorders $\widehat{\sqsubseteq}_{\text{pmay}}$ and $\widehat{\sqsubseteq}_{\text{pmust}}$ also coincide with $\sqsubseteq_{\text{pmay}}$ and $\sqsubseteq_{\text{pmust}}$.[1]

Simulation preorders: In Section 5.5 we use the transitions $s \xrightarrow{\alpha} \Delta$ to define two co-inductive preorders, the *simulation* preorder $\sqsubseteq_S$ [Seg95, LSV03, DvGH+07a], and the novel *failure simulation* preorder $\sqsubseteq_{FS}$ over pCSP processes. The latter extends the failure simulation preorder of [Gla93] to

[1] However in the presence of divergence they are slightly different.

probabilistic processes. Their definition uses a natural generalisation of the transitions, first (Kleisli-style) to take the form $\Delta \xrightarrow{\alpha} \Delta'$, and then to *weak* versions $\Delta \stackrel{\alpha}{\Longrightarrow} \Delta'$. The second preorder differs from the first one in the use of a *failure* predicate $s \stackrel{X}{\not\longrightarrow}$, indicating that in the state s none of the actions in X can be performed.

Both preorders are preserved by all the operators in pCSP, and are *sound* with respect to the testing preorders; that is $P \sqsubseteq_S Q$ implies $P \sqsubseteq_{\text{pmay}} Q$ and $P \sqsubseteq_{FS} Q$ implies $P \sqsubseteq_{\text{pmust}} Q$. For $\sqsubseteq_S$ this was established in [DvGH$^+$07a], and the proofs for $\sqsubseteq_{FS}$ are similar. But *completeness*, that the testing preorders imply the respective simulation preorders, requires some ingenuity. We prove it indirectly, involving a characterisation of the testing and simulation preorders in terms of a modal logic.

Modal logic: Our modal logic, defined in Section 5.9, uses finite conjunction $\bigwedge_{i\in I} \phi_i$, the modality $\langle a\rangle\phi$ from the Hennessy-Milner Logic [HM85], and a novel probabilistic construct $\bigoplus_{i\in I} p_i \cdot \phi_i$. A satisfaction relation between processes and formulae then gives, in a natural manner, a *logical preorder* between processes: $P \sqsubseteq^{\mathcal{L}} Q$ means that every $\mathcal{L}$-formula satisfied by P is also satisfied by Q. We establish that $\sqsubseteq^{\mathcal{L}}$ coincides with $\sqsubseteq_S$ (and hence $\sqsubseteq_{\text{pmay}}$ also).

To capture failures, we add, for every set of actions X, a formula $\mathbf{ref}(X)$ to our logic, satisfied by any process which, after it can do no further internal actions, can perform none of the actions in X either. The constructs $\bigwedge$, $\langle a\rangle$ and $\mathbf{ref}()$ stem from the modal characterisation of the non-probabilistic failure simulation preorder, given in [Gla93]. We show that $\sqsubseteq_{\text{pmust}}$, as well as $\sqsubseteq_{FS}$, can be characterised in a similar manner with this extended modal logic.

Proof strategy: We prove these characterisation results through two cycles of inclusions:

$$\underbrace{\sqsubseteq^{\mathcal{L}} \subseteq}_{\text{Sec. 5.9}} \quad \underbrace{\sqsubseteq_S \overset{[\text{DvGH}^+07a]}{\subseteq} \sqsubseteq_{FS}}_{\text{Sec. 5.5}} \quad \underbrace{\sqsubseteq_{\text{pmay}} \subseteq \sqsubseteq_{\text{pmust}}}_{\text{Sec. 4.6.1}} \quad \underbrace{\subseteq \widehat{\sqsubseteq}_{\text{pmay}} \subseteq \widehat{\sqsubseteq}_{\text{pmust}}}_{\text{Sec. 5.6}} \quad \underbrace{\overset{[\text{DvGMZ07}]}{=} \widehat{\sqsubseteq}^{\Omega}_{\text{pmay}} \overset{[\text{DvGMZ07}]}{=} \widehat{\sqsubseteq}^{\Omega}_{\text{pmust}}}_{\text{Sec. 5.7}} \quad \underbrace{\subseteq \sqsubseteq^{\mathcal{L}} \subseteq \sqsubseteq^{\mathcal{F}}}_{\text{Sec. 5.10}}$$

In Section 5.9 we show that $P \sqsubseteq^{\mathcal{L}} Q$ implies $P \sqsubseteq_S Q$ (and hence $P \sqsubseteq_{\text{pmay}} Q$), and likewise for $\sqsubseteq^{\mathcal{F}}$ and $\sqsubseteq_{FS}$; the proof involves constructing, for each pCSP process P, a *characteristic formula* ϕ_P. To obtain the other direction, in Section 5.10 we show how every modal formula ϕ can be captured, in some sense, by a test T_ϕ; essentially the ability of a pCSP process to satisfy ϕ is determined by its ability to pass the test T_ϕ. We capture the conjunction of two formulae by a probabilistic choice between the corresponding tests; in order to prevent the results from these tests getting mixed up, we employ the vector-based tests of [Seg96], so that we can use different success actions in the separate probabilistic branches. Therefore, we complete our proof by demonstrating that the state-based testing preorders imply the action-based ones (Section 5.6) and recalling Theorem 4.64 that the action-based scalar testing preorders imply the vector-based ones (Section 5.7).

(In)equations: It is well-known that may- and must testing for standard CSP can be captured equationally [DNH84, BHR84, Hen88]. In Section 5.3 we show that most of the standard equations are no longer valid in the probabilistic setting of pCSP. However, we show in Section 5.12 that both $P \sqsubseteq_{\text{pmay}} Q$ and $P \sqsubseteq_{\text{pmust}} Q$ can still be captured equationally over full pCSP. In the may case the essential (in)equation required is

$$a.(P \oplus_p Q) \quad \sqsubseteq \quad a.P \oplus_p a.Q$$

The must case is more involved: in the absence of the distributivity of the external and internal choices over each other, to obtain completeness we require a complicated inequational schema.

5.2 Finite probabilistic CSP

We first define the language and its operational semantics. Then we show how the general probabilistic testing theory outlined in Section 4.6.1 can be applied to processes from this language.

5.2.1 The language

Let Act be a set of actions, ranged over by $a, b, c, \ldots$, which processes can perform. Then the finite probabilistic CSP processes are given by the following syntax:

$$P ::= \mathbf{0} \mid a.P \mid P \sqcap P \mid P \,\Box\, P \mid P \mid_A P \mid P \,{}_p\!\oplus\, P$$

The intuitive meaning of the various constructs is straightforward:

(i) $\mathbf{0}$ represents the stopped process.

(ii) $a.P$, where a is in Act, is a process which first performs the action a, and then proceeds as P.

(iii) $P \sqcap Q$ is the *internal choice* between the processes P and Q; it will act either like P or like Q, but a user is unable to influence which.

(iv) $P \,\Box\, Q$ is the *external choice* between P and Q; again it will act either like P or like Q but, in this case, according to the demands of a user.

(v) $P \mid_A Q$, where A is a subset of Act, represents processes P and Q running in parallel. They may synchronise by performing the same action from A simultaneously; such a synchronisation results in an internal action $\tau \notin Act$. In addition P and Q may independently do any action from $(Act \setminus A) \cup \{\tau\}$.

(vi) $P \,{}_p\!\oplus\, Q$, where p is an arbitrary probability, a real number strictly between 0 and 1, is the *probabilistic choice* between P and Q. With probability p it will act like P and with probability $(1-p)$ it will act like Q.

We use pCSP to denote the set of terms defined by this grammar, and CSP denotes the subset of that which does not use the probabilistic choice. Of course the language CSP in all its glory [BHR84, Hoa85b, OH86] has many more operators; we have simply chosen a representative selection, adding probabilistic choice to obtain an elementary probabilistic version of CSP. Our parallel operator is not a CSP primitive, but it can easily be expressed in terms of the CSP primitives—in particular $P \mid_A Q = (P \| _A Q) \backslash A$, where $\|_A$ and $\backslash A$ are the parallel composition and hiding operators of [OH86]. It can also be expressed in terms of the parallel composition, renaming and restriction operators of CCS. We have chosen this (non-associative) operator for its convenience in defining the application of tests to processes.

As usual we tend to omit occurrences of $\mathbf{0}$; the action prefixing operator $a.\_$ binds stronger than any of the binary operators, and precedence between the binary operators will be indicated via brackets or spacing. We will also sometimes use n-ary versions of the binary operators, such as $\bigoplus_{i\in I} p_i P_i$ with $\sum_{i\in I} p_i = 1$, and $\bigsqcap_{i\in I} P_i$.

5.2.2 Operational semantics of pCSP

The above intuitive reading of the various operators can be formalised by an *operational semantics* which associates with each process term a graph-like structure representing the manner in which it may react to users' requests. Let us briefly recall this procedure for (non-probabilistic) CSP.

The operational semantics of CSP is obtained by endowing the set of terms with the structure of an LTS (cf. Section 2.1). Specifically

(i) the set of states S is taken to be all terms from the language CSP

(ii) the action relations $P \xrightarrow{\alpha} Q$ are defined inductively on the syntax of terms.

A precise definition may be found in [OH86].

In order to interpret the full pCSP operationally we need to use pLTSs, the probabilistic generalisation of LTSs (see Section 4.1).

We now mimic the operational interpretation of CSP as an LTS by associating with pCSP a particular pLTS $\langle S_p, Act_\tau, \rightarrow \rangle$. However there are two major differences:

(i) only a subset of terms in pCSP will be used as the set of states S_p in the pLTS

(ii) terms in pCSP will be interpreted as distributions over S_p, rather than as elements of S_p.

First we define the subset S_p of states that we use: it is the least set satisfying

(i) $\mathbf{0} \in S_p$

(ii) $a.P \in S_p$

(iii) $P \sqcap Q \in S_p$

(iv) $s_1, s_2 \in S_p$ implies $s_1 \,\Box\, s_2 \in S_p$

(v) $s_1, s_2 \in S_p$ implies $s_1 \mid_A s_2 \in S_p$.

Thus, S_p is the set of pCSP expressions in which every occurrence of the probabilistic choice operator ${}_p\!\oplus$ is *weakly guarded*, i.e. is within a subexpression of the form $a.P$ or $P \sqcap Q$. The interpretation of terms in pCSP as distributions in $\mathcal{D}(S_p)$ is as follows:

1. $[\![\mathbf{0}]\!] = \overline{\mathbf{0}}$
2. $[\![a.P]\!] = \overline{a.P}$
3. $[\![P \sqcap Q]\!] = \overline{P \sqcap Q}$
4. $[\![P \,{}_p\!\oplus\, Q]\!] = p \cdot [\![P]\!] + (1-p) \cdot [\![Q]\!]$
5. $[\![P \,\Box\, Q]\!] = [\![P]\!] \,\Box\, [\![Q]\!]$
6. $[\![P \mid_A Q]\!] = [\![P]\!] \mid_A [\![Q]\!]$.

In the last two cases we take advantage of the fact that both $\Box$ and $\mid_A$ can be viewed as binary operators over S_p, and can therefore be lifted to $\mathcal{D}(S_p)$ in the standard manner. Namely we define

$$(\Delta_1 \,\Box\, \Delta_2)(s) \quad = \quad \begin{cases} \Delta_1(t_1) \cdot \Delta_2(t_2) & \text{if } s = t_1 \,\Box\, t_2, \\ 0 & \text{otherwise} \end{cases}$$

with $\Delta_1 \mid_A \Delta_2$ given similarly; note that as a result we have $[\![P]\!] = \overline{P}$ for all $P \in S_p$. Finally the definition of the relations $\xrightarrow{\alpha}$ is given in Figure 5.1. These rules are very similar to the standard ones used to interpret CSP as an LTS [OH86], modified to take into account that the result of an action must be a distribution. For example (INT.L) and (INT.R) say that $P \sqcap Q$ makes an internal unobservable choice to act either like P or like Q. Similarly the four rules (EXT.L), (EXT.R), (EXT.I.L) and (EXT.I.R) can be read as giving the standard interpretation to the external choice operator: the process $P \,\Box\, Q$ may perform any external action of its components P and Q, which resolves the choice; it may also perform any of their internal actions, but when these are performed the choice is not resolved.

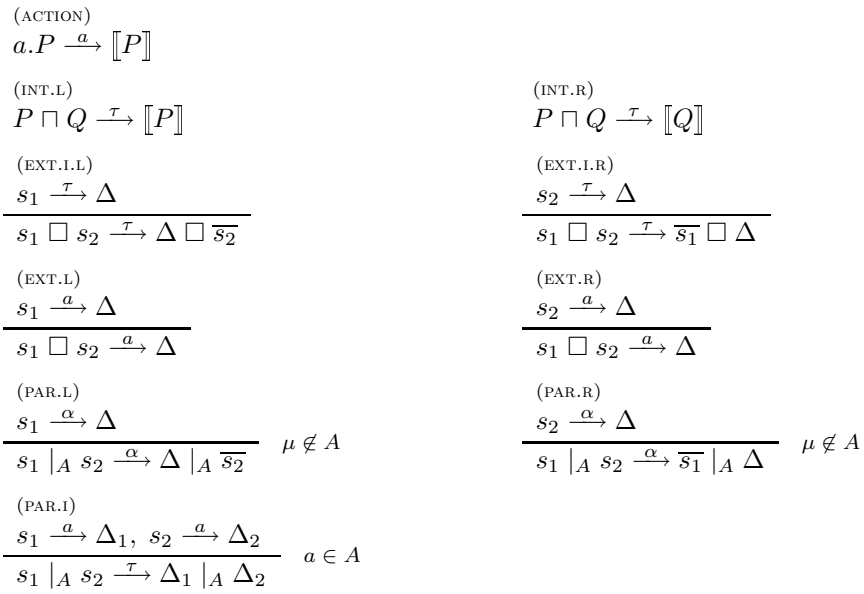


Figure 5.1: Operational semantics of pCSP. Here a ranges over Act and μ over Act_τ.

5.2.3 The precedence of probabilistic choice

Our operational semantics entails that $\square$ and $|_A$ distribute over probabilistic choice:

$$\begin{array}{rcl} [\![P \,\square\, (Q \,{}_p\!\oplus\, R)]\!] & = & [\![(P \,\square\, Q) \,{}_p\!\oplus\, (P \,\square\, R)]\!] \\ [\![P \,|_A\, (Q \,{}_p\!\oplus\, R)]\!] & = & [\![(P \,|_A\, Q) \,{}_p\!\oplus\, (P \,|_A\, R)]\!] \end{array}$$

In Section 5.11, these identities will return as axioms of may testing. However, this is not so much a consequence of our testing methodology: it is hardwired in our interpretation $[\![\_]\!]$ of pCSP expressions as distributions. We could have obtained the same effect by introducing pCSP as a two-sorted language, given by the grammar

$$\begin{array}{rcl} P & ::= & S \mid P \,{}_p\!\oplus\, P \\ S & ::= & \mathbf{0} \mid a.P \mid P \sqcap P \mid S \,\square\, S \mid S \,|_A\, S \end{array}$$

and introducing expressions like $P \,\square\, (Q \,{}_p\!\oplus\, R)$ and $P \,|_A\, (Q \,{}_p\!\oplus\, R)$ as syntactic sugar for the grammatically correct expressions obtained by distributing $\square$ and $|_A$ over ${}_p\!\oplus$. In that case, the S-expressions would constitute the set S_p of states in the pLTS of pCSP, and $[\![s]\!] = \overline{s}$ for any $s \in S_p$.

A consequence of our operational semantics is that in the process $a.(b \,{}_{\frac{1}{2}}\!\oplus\, c) \,|_\emptyset\, d$ the action d can be scheduled either before a, or after the probabilistic choice between b and c—but it can *not* be scheduled after a and before this probabilistic choice. We justify this by thinking of $P \,{}_p\!\oplus\, Q$ not as a process that starts with making a probabilistic choice, but rather as one that *has* just made such a choice, and with probability p *is* no more and no less than the process P. Thus $a.(P \,{}_p\!\oplus\, Q)$ is a process that in doing the a-step makes a probabilistic choice between the alternative targets P and Q.

This design decision is in full agreement with previous work featuring nondeterminism, probabilistic choice and parallel composition [HJ90, YL92, Seg95]. Moreover, a probabilistic choice between processes P and Q that does not take precedence over actions scheduled in parallel can simply be written as $\tau.(P \,{}_p\!\oplus\, Q)$. Here $\tau.P$ is an abbreviation for $P \sqcap P$. Using the operational semantics of $\sqcap$ in Figure 5.1, $\tau.P$ is a process whose sole initial transition is $\tau.P \xrightarrow{\tau} P$, hence $\tau.(P \,{}_p\!\oplus\, Q)$ is a process that starts with making a probabilistic choice, modelled as an internal action, and with probability p proceeds as P. Any activity scheduled in parallel with $\tau.(P \,{}_p\!\oplus\, Q)$ can now be scheduled

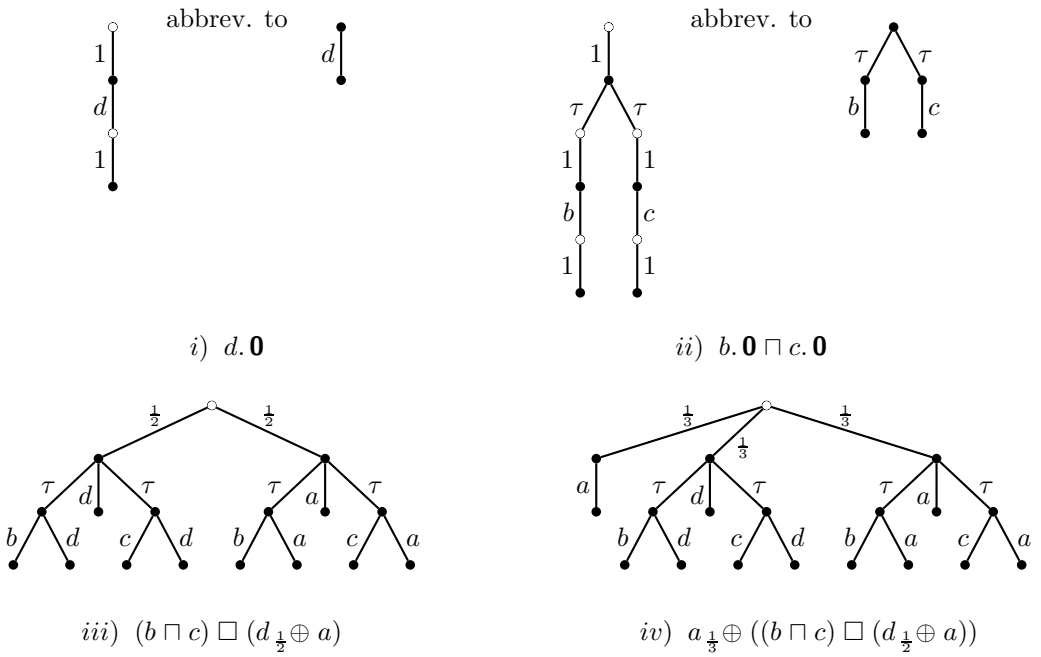


Figure 5.2: Example pLTSs

before or after this internal action, hence before or after the making of the choice. In particular, $a.\tau.(b \,_{\frac{1}{2}}\!\oplus c) \mid_{\emptyset} d$ allows d to happen between a and the probabilistic choice.

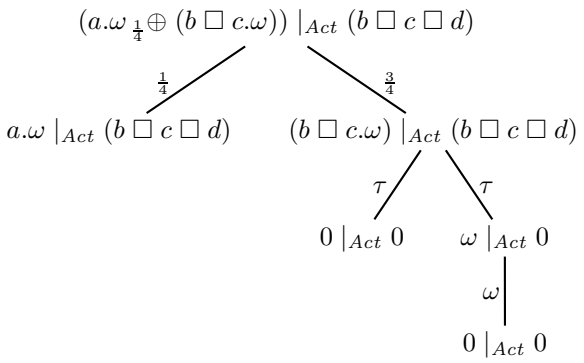
5.2.4 Graphical representation of pCSP processes

We graphically depict the operational semantics of a pCSP expression P by drawing the part of the pLTS defined above that is reachable from $[\![P]\!]$ as a finite acyclic directed graph. Given that in a pLTS transitions go from states to distributions, we need to introduce additional edges to connect distributions back to states, thereby obtaining a bipartite graph. States are represented by nodes of the form $\bullet$ and distributions by nodes of the form $\circ$. For any state s and distribution Δ with $s \xrightarrow{\alpha} \Delta$ we draw an edge from s to Δ, labelled with α. Consequently, the edges leaving a $\bullet$-node are all labelled with actions from Act_τ. For any distribution Δ and state s in $\lceil\Delta\rceil$, the support of Δ, we draw an edge from Δ to s, labelled with $\Delta(s)$. Consequently, the edges leaving a $\circ$-node are labelled with positive real numbers that sum up to 1. Because for our simple language the resulting directed bipartite graphs are acyclic, we leave out arrowheads on edges and we assume control to flow from top to bottom.

A few examples are described in Figure 5.2. In *i*) we find $[\![d.\mathbf{0}]\!]$ as the point distribution $\overline{d.\mathbf{0}}$, represented by a tree with one edge from the root, labelled with the probability 1, to the state $d.\mathbf{0}$. In turn this state is represented as the subtree with one outgoing edge, labelled by the only possible action d to $[\![\mathbf{0}]\!]$. Finally this is also a point distribution, represented as a subtree with one edge leading to a leaf, labelled by the probability 1.

These edges labelled by probability 1 occur so frequently that it makes sense to omit them, together with the associated nodes $\circ$ representing point distributions. With this convention $[\![d.\mathbf{0}]\!]$ turns out to be a simple tree with one edge labelled by the action d. The same convention simplifies considerably the representation of $b \sqcap c$ in *ii*), resulting in an LTS detailing an internal choice between the two actions. Officially, the endnodes of this graph ought to be merged, as both of them represent the process $\mathbf{0}$. However, for reasons of clarity, in graphical representations we often unwind the underlying graph into a tree, thus representing the same state or distribution multiple times.

The interpretation of $(b \sqcap c) \,\square\, (d \,_{\frac{1}{2}}\!\oplus a)$ is more interesting. This requires clause (v) above in the



$$\mathcal{A}pply((a.\omega \,{}_{\frac{1}{4}}\!\oplus (b \,\Box\, c.\omega)), (b \,\Box\, c \,\Box\, d)) = \tfrac{1}{4} \cdot \{0\} + \tfrac{3}{4} \cdot \{0, 1\} = \{0, \tfrac{3}{4}\}$$

Figure 5.3: Example of testing

definition of $[\![\_]\!]$, resulting in the distribution $\overline{(b \sqcap c)} \,\Box\, \Delta$, where Δ is the distribution resulting from the interpretation of $(d \,{}_{\frac{1}{2}}\!\oplus\, a)$, itself a two-point distribution mapping both the states $d.\,\mathbf{0}$ and $a.\,\mathbf{0}$ to the probability $\frac{1}{2}$. The result is again a two-point distribution, this time mapping the two states $(b \sqcap c) \,\Box\, d$ and $(b \sqcap c) \,\Box\, a$ to $\frac{1}{2}$. The end result in *iii*) is obtained by further interpreting these two states using the rules in Figure 5.1. Finally in *iv*) we show the graphical representation which results when this term is combined probabilistically with the simple process $a.\,\mathbf{0}$.

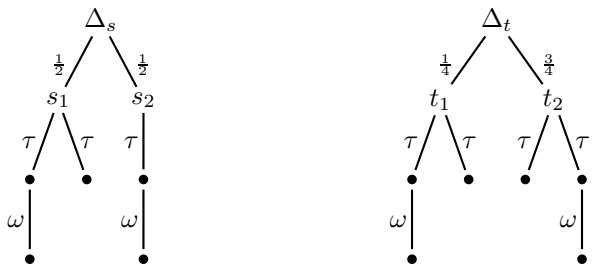
To sum up, the operational semantics endows pCSP with the structure of a pLTS, and the function $[\![\_]\!]$ interprets process terms in pCSP as distributions in this pLTS, which can be represented by finite acyclic directed graphs (typically drawn as trees), with edges labelled either by probabilities or actions, so that edges labelled by actions and probabilities alternate (although in pictures we may suppress 1-labelled edges and point distributions).

5.2.5 Testing pCSP processes

Let us now turn to applying the testing theory from Section 4.6.1 to pCSP. As with the standard theory [DNH84, Hen88], we use as tests any process from pCSP itself, which in addition can use a special symbol ω to report success. For example, $a.\omega \,{}_{\frac{1}{4}}\!\oplus (b \,\Box\, c.\omega)$ is a probabilistic test, which 25% of the time requests an a action, and 75% requests that c can be performed. If it is used as *must* test the 75% that requests a c action additionally requires that b is not possible. As in [DNH84, Hen88], it is not the execution of ω that constitutes success, but the arrival in a state where ω is possible. The introduction of the ω-action is simply a method of defining a success predicate on states without having to enrich the language of processes explicitly with such predicates.

Formally, let $\omega \notin Act_\tau$ and write Act^ω for $Act \cup \{\omega\}$ and Act^ω_τ for $Act \cup \{\tau, \omega\}$. In Figure 5.1 we now let a range over Act^ω and α over Act^ω_τ. Tests may have subterms $\omega.P$, but other processes may not. To apply the test T to the process P we run them in parallel, tightly synchronised; that is, we run the combined process $T \,|_{Act}\, P$. Here P can only synchronise with T, and in turn the test T can only perform the success action ω, in addition to synchronising with the process being tested; of course both tester and testee can also perform internal actions. An example is provided in Figure 5.3, where the test $a.\omega \,{}_{\frac{1}{4}}\!\oplus (b \,\Box\, c.\omega)$ is applied to the process $b \,\Box\, c \,\Box\, d$. We see that 25% of the time the test is unsuccessful, in that it does not reach a state where ω is possible, and 75% of the time it *may* be successful, depending on how the now internal choice between the actions b and c is resolved, but it is not the case that it *must* be successful.

$[\![T \,|_{Act}\, P]\!]$ is representable as a finite graph which encodes all possible interactions of the test T with the process P. It only contains the actions τ and ω. Each occurrence of τ represents a



$\mathbb{V}(\Delta_s) = \{\frac{1}{2},\ 1\}$ $\mathbb{V}(\Delta_t) = \{0,\ \frac{1}{4},\ \frac{3}{4},\ 1\}$

Figure 5.4: Collecting results

nondeterministic choice, either in T or P themselves, or as a nondeterministic response by P to a request from T, while the distributions represent the resolution of underlying probabilities in T and P. We use the structure $[\![T \mid_{Act} P]\!]$ to define $\mathcal{A}pply(T, P)$, the non-empty finite subset of $[0, 1]$ representing the set of probabilities that applying T to P will be a success.

First we define a function $\mathbb{V} : S_p \to \mathcal{P}^+_{fin}([0, 1])$, which when applied to any state in S_p returns a finite subset of $[0, 1]$; it extends to type $\mathcal{D}(S_p) \to \mathcal{P}^+_{fin}([0, 1])$ via the convention $\mathbb{V}(\Delta) := \mathrm{Exp}_\Delta \mathbb{V}$ (cf. Section 4.1).

$$\mathbb{V}(s) \quad = \quad \begin{cases} \{1\} & \text{if } s \xrightarrow{\omega}, \\ \bigcup\{\, \mathbb{V}(\Delta) \mid s \xrightarrow{\tau} \Delta \,\} & \text{if } s \not\xrightarrow{\omega},\ s \xrightarrow{\tau}, \\ \{0\} & \text{otherwise} \end{cases}$$

We will tend to write the expected value of $\mathbb{V}$ explicitly and use the convenient notation

$$\mathbb{V}(\Delta) \quad = \quad \Delta(s_1) \cdot \mathbb{V}(s_1) + \ldots + \Delta(s_n) \cdot \mathbb{V}(s_n)$$

where $\lceil \Delta \rceil = \{s_1, \ldots s_n\}$. Note that $\mathbb{V}(\_)$ is indeed a well-defined function, because the pLTS $\langle S_p, Act_\tau, \to \rangle$ is finitely branching and well-founded.

For example consider the transition systems in Figure 5.4, where for reference we have labelled the nodes. Then $\mathbb{V}(s_1) = \{1, 0\}$ while $\mathbb{V}(s_2) = \{1\}$, and therefore $\mathbb{V}(\Delta_s) = \frac{1}{2} \cdot \{1, 0\} + \frac{1}{2} \cdot \{1\}$ which, since there are only two possible choice functions $c \in_{\lceil \Delta_s \rceil} \mathbb{V}$, evaluates further to $\{\frac{1}{2}, 1\}$. Similarly $\mathbb{V}(t_1) = \mathbb{V}(t_2) = \{0, 1\}$ and using the four choice functions $c \in_{\lceil \Delta_t \rceil} \mathbb{V}$, the calculation of $\mathbb{V}(\Delta_t) = \frac{1}{4} \cdot \{0, 1\} + \frac{3}{4} \cdot \{0, 1\}$ leads to $\{0, \frac{1}{4}, \frac{3}{4}, 1\}$.

Definition 5.1 *For any $P \in$ pCSP and $T \in \mathcal{T}$ let $\mathcal{A}pply(T, P) = \mathbb{V}([\![T \mid_{Act} P]\!])$.*

With this definition we now have two testing preorders for pCSP, one based on *may testing*, $P \sqsubseteq_{\mathrm{pmay}} Q$, and the other on *must testing*, $P \sqsubseteq_{\mathrm{pmust}} Q$.

5.3 Counterexamples

We will see in this section that many of the standard testing axioms are not valid in the presence of probabilistic choice. We also provide counterexamples for a few distributive laws involving probabilistic choice that may appear plausible at first sight. In all cases we establish a statement $P \not\simeq_{\mathrm{pmay}} Q$ by exhibiting a test T such that $\max(\mathcal{A}pply(T, P)) \neq \max(\mathcal{A}pply(T, Q))$ and a statement $P \not\simeq_{\mathrm{pmust}} Q$ by exhibiting a test T such that $\min(\mathcal{A}pply(T, P)) \neq \min(\mathcal{A}pply(T, Q))$. In case $\max(\mathcal{A}pply(T, P)) > \max(\mathcal{A}pply(T, Q))$ we find in particular that $P \not\sqsubseteq_{\mathrm{pmay}} Q$, and in case $\min(\mathcal{A}pply(T, P)) > \min(\mathcal{A}pply(T, Q))$ we obtain $P \not\sqsubseteq_{\mathrm{pmust}} Q$.

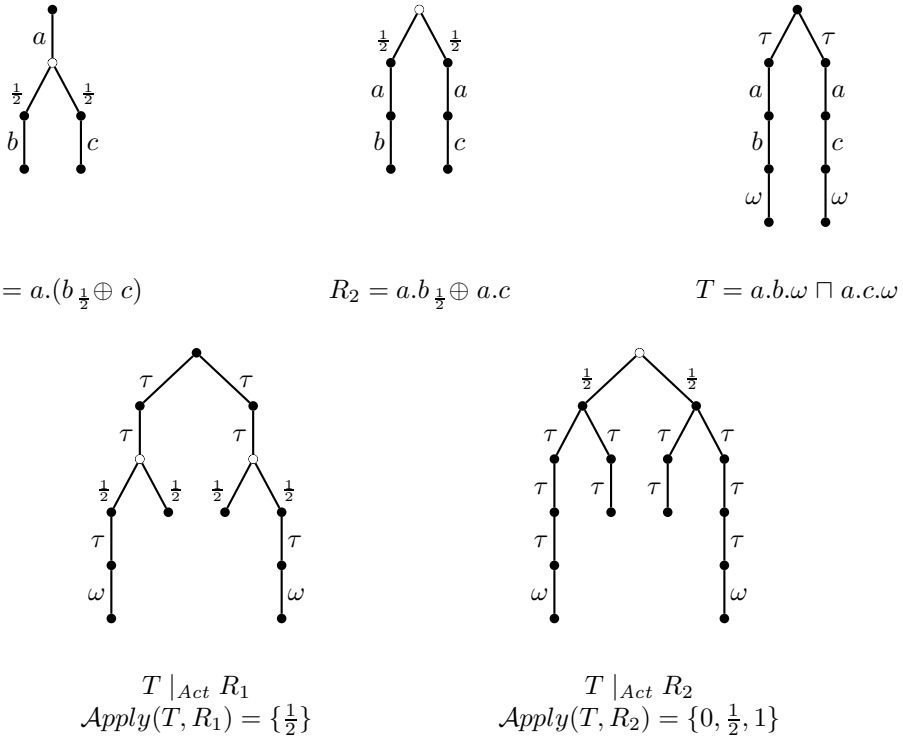


Figure 5.5: Counterexample: action prefix does not distribute over probabilistic choice

Example 5.2 *The axiom $a.(P\ {}_p\!\oplus\ Q) = a.P\ {}_p\!\oplus\ a.Q$ is unsound.*

Consider the example in Figure 5.5. In R_1 the probabilistic choice between b and c is taken after the action a, while in R_2 the choice is made before the action has happened. These processes can be distinguished by the nondeterministic test $T = a.b.\omega \sqcap a.c.\omega$. First consider running this test on R_1. There is an immediate choice made by the test, effectively running either the test $a.b.\omega$ on R_1 or the test $a.c.\omega$; in fact the effect of running either test is exactly the same. Consider $a.b.\omega$. When run on R_1 the a action immediately happens, and there is a probabilistic choice between running $b.\omega$ on either b or c, giving as possible results $\{1\}$ or $\{0\}$; combining these according to the definition of the function $\mathbb{V}(\_)$ we get $\frac{1}{2}\cdot\{0\} + \frac{1}{2}\cdot\{1\} = \{\frac{1}{2}\}$. Since running the test $a.c.\omega$ has the same effect, $\mathcal{A}pply(T, R_1)$ turns out to be the same set $\{\frac{1}{2}\}$.

Now consider running the test T on R_2. Because R_2, and hence also $T \mid_{Act} R_2$, starts with a probabilistic choice, due to the definition of the function $\mathbb{V}(\_)$, the test must first be applied to the probabilistic components, $a.b$ and $a.c$, respectively, and the results subsequently combined probabilistically. When the test is run on $a.b$, immediately a nondeterministic choice is made in the test, to run either $a.b.\omega$ or $a.c.\omega$. With the former we get the result $\{1\}$, with the latter $\{0\}$, so overall, for running T on $a.b$, we get the possible results $\{0, 1\}$. The same is true when we run it on $a.c$, and therefore $\mathcal{A}pply(T, R_2) = \frac{1}{2}\cdot\{0,1\} + \frac{1}{2}\cdot\{0,1\} = \{0, \frac{1}{2}, 1\}$.

So we have $R_2 \not\sqsubseteq_{pmay} R_1$ and $R_1 \not\sqsubseteq_{pmust} R_2$.

Example 5.3 *The axiom $a.(P \sqcap Q) = a.P \sqcap a.Q$ is unsound.*

It is well known that this axiom is valid in the standard theory of testing, for non-probabilistic processes. However, consider the instance R_1 and R_2 in Figure 5.6, and note that these processes do not contain any probabilistic choice. But they can be differentiated by the probabilistic test $T = a.(b.\omega\ {}_{\frac{1}{2}}\!\oplus\ c.\omega)$; the details are in Figure 5.6. There is only one possible outcome from ap-

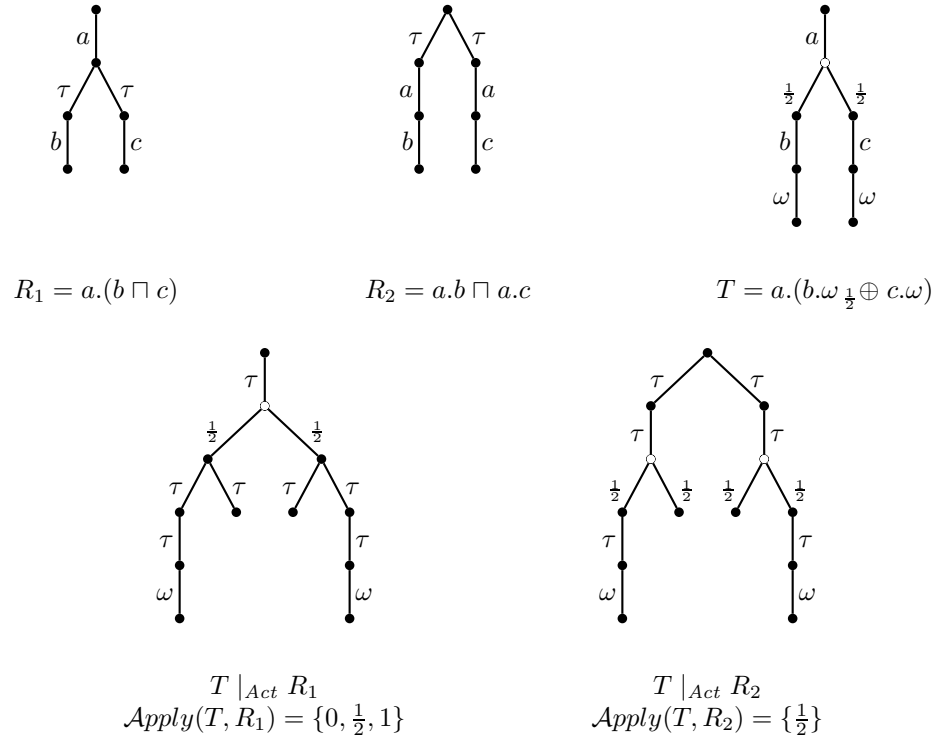


Figure 5.6: Counterexample: action prefix does not distribute over internal choice

plying T to R_2, the probability $\frac{1}{2}$, because the nondeterministic choice is made before the probabilistic choice. On the other hand when T is applied to R_1 there are three possible outcomes, 0, $\frac{1}{2}$ and 1, because effectively the probabilistic choice takes precedence over the nondeterministic choice. So we have $R_1 \not\sqsubseteq_{pmay} R_2$ and $R_2 \not\sqsubseteq_{pmust} R_1$.

Example 5.4 *The axiom $a.(P \,\square\, Q) = a.P \,\square\, a.Q$ is unsound.*

This axiom is valid in the standard may-testing semantics. However, consider the two processes $R_1 = a.(b \,\square\, c)$, $R_2 = a.b \,\square\, a.c$ and the probabilistic test $T = a.(b.\omega \,{}_{\frac{1}{2}}\!\oplus c.\omega)$. Now $\mathcal{A}pply(T, R_1) = \{1\}$ and $\mathcal{A}pply(T, R_2) = \{\frac{1}{2}\}$. Therefore $R_1 \not\sqsubseteq_{pmay} R_2$ and $R_1 \not\sqsubseteq_{pmust} R_2$.

Example 5.5 *The axiom $P = P \,\square\, P$ is unsound.*

Let R_1, R_2 denote $(a \,{}_{\frac{1}{2}}\!\oplus\, b)$ and $(a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\square\, (a \,{}_{\frac{1}{2}}\!\oplus\, b)$, respectively. It is easy to calculate that $\mathcal{A}pply(a.\omega, R_1) = \{\frac{1}{2}\}$ but, because of the way we interpret external choice as an operator over distributions of states in a pLTS, it turns out that $[\![R_2]\!] = [\![((a \,\square\, a) \,{}_{\frac{1}{2}}\!\oplus\, (a \,\square\, b)) \,{}_{\frac{1}{2}}\!\oplus\, ((b \,\square\, a) \,{}_{\frac{1}{2}}\!\oplus\, (b \,\square\, b))]\!]$ and so $\mathcal{A}pply(a.\omega, R_2) = \{\frac{3}{4}\}$. Therefore $R_2 \not\sqsubseteq_{pmay} R_1$ and $R_2 \not\sqsubseteq_{pmust} R_1$.

Example 5.6 *The axiom $P \,{}_{p}\!\oplus\, (Q \sqcap R) = (P \,{}_{p}\!\oplus\, Q) \sqcap (P \,{}_{p}\!\oplus\, R)$ is unsound.*

Consider the processes $R_1 = a \,{}_{\frac{1}{2}}\!\oplus\, (b \sqcap c)$ and $R_2 = (a \,{}_{\frac{1}{2}}\!\oplus\, b) \sqcap (a \,{}_{\frac{1}{2}}\!\oplus\, c)$, and the test $T_1 = a.\omega \sqcap (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega)$. In the best of possible worlds, when we apply T_1 to R_1 we obtain probability 1, that is $\max(\mathcal{A}pply(T_1, R_1)) = 1$. Informally this is because half the time when it is applied to the subprocess a of R_1, optimistically the sub-test $a.\omega$ is actually run. The other half of the time, when it is applied to the subprocess $(b \sqcap c)$, optimistically the sub-test $T_r = (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega)$ is actually used. And here again, optimistically, we obtain probability 1: whenever the test $b.\omega$ is used it might be applied to the subprocess b, while when $c.\omega$ is used it might be applied to c. Formally we have

$$\begin{aligned}\mathcal{A}pply(T_1, R_1) &= \tfrac{1}{2} \cdot \mathcal{A}pply(T_1, a) + \tfrac{1}{2} \cdot \mathcal{A}pply(T_1, b \sqcap c) \\ &= \tfrac{1}{2} \cdot (\mathcal{A}pply(a.\omega, a) \cup \mathcal{A}pply(T_r, a)) + \\ &\tfrac{1}{2} \cdot (\mathcal{A}pply(T_1, b) \cup \mathcal{A}pply(T_1, c) \cup \mathcal{A}pply(a.\omega, b \sqcap c) \cup \mathcal{A}pply(T_r, b \sqcap c)) \\ &= \tfrac{1}{2} \cdot (\{1\} \cup \{0\}) + \tfrac{1}{2} \cdot (\{0, \tfrac{1}{2}\} \cup \{0, \tfrac{1}{2}\} \cup \{0\} \cup \{0, \tfrac{1}{2}, 1\}) \\ &= \{0, \tfrac{1}{4}, \tfrac{1}{2}, \tfrac{3}{4}, 1\}\end{aligned}$$

However no matter how optimistic we are when applying T_1 to R_2 we can never get probability 1; the most we can hope for is $\frac{3}{4}$, which might occur when T_1 is applied to the subprocess $(a \,{}_{\frac{1}{2}}\!\oplus\, b)$. Specifically when the subprocess a is being tested the sub-test $a.\omega$ might be used, giving probability 1, and when the subprocess b is being tested the sub-test $(b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega)$ might be used, giving probability $\frac{1}{2}$. We leave the reader to check that formally

$$\mathcal{A}pply(T_1, R_2) = \{0, \tfrac{1}{4}, \tfrac{1}{2}, \tfrac{3}{4}\}$$

from which we can conclude $R_1 \not\sqsubseteq_{pmay} R_2$.

We can also show that $R_2 \not\sqsubseteq_{pmust} R_1$, using the test

$$T_2 = (b.\omega \,\Box\, c.\omega) \sqcap (a.\omega \,{}_{\frac{1}{3}}\!\oplus\, (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega)).$$

Reasoning pessimistically, the worst that can happen when applying T_2 to R_1 is we get probability 0. Each time the subprocess a is tested the worst probability will occur when the sub-test $(b.\omega \,\Box\, c.\omega)$ is used; this results in probability 0. Similarly when the subprocess $(b \sqcap c)$ is being tested the subtest $(a.\omega \,{}_{\frac{1}{3}}\!\oplus\, (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega))$ will give probability 0. In other words $\min(\mathcal{A}pply(T_2, R_1)) = 0$. When applying T_2 to R_2, things can never be as bad. The worst probability will occur when T_2 is applied to the subprocess $(a \,{}_{\frac{1}{2}}\!\oplus\, b)$, namely probability $\frac{1}{6}$. We leave the reader to check that formally $\mathcal{A}pply(T_2, R_1) = \{0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$ and $\mathcal{A}pply(T_2, R_2) = \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$.

Example 5.7 *The axiom $P \sqcap (Q \,{}_{p}\!\oplus\, R) = (P \sqcap Q) \,{}_{p}\!\oplus\, (P \sqcap R)$ is unsound.*

Let $R_1 = a \sqcap (b \,{}_{\frac{1}{2}}\!\oplus\, c)$, $R_2 = (a \sqcap b) \,{}_{\frac{1}{2}}\!\oplus\, (a \sqcap c)$ and $T = a.(\omega \,{}_{\frac{1}{2}}\!\oplus\, \mathbf{0}) \,\Box\, b.\omega$. One can check that $\mathcal{A}pply(T, R_1) = \{\frac{1}{2}\}$ and $\mathcal{A}pply(T, R_2) = \frac{1}{2}\{\frac{1}{2}, 1\} + \frac{1}{2}\{\frac{1}{2}, 0\} = \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. Therefore we have $R_2 \not\sqsubseteq_{pmay} R_1$ and $R_1 \not\sqsubseteq_{pmust} R_2$.

Example 5.8 *The axiom $P \,\Box\, (Q \sqcap R) = (P \,\Box\, Q) \sqcap (P \,\Box\, R)$ is unsound.*

Let $R_1 = (a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\Box\, (c \sqcap d)$, $R_2 = ((a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\Box\, c) \sqcap ((a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\Box\, d)$ and $T = (a.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega) \sqcap (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, d.\omega)$. This time we get $\mathcal{A}pply(T, R_1) = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ and $\mathcal{A}pply(T, R_2) = \{\frac{1}{4}, \frac{3}{4}\}$. So $R_1 \not\sqsubseteq_{pmay} R_2$ and $R_2 \not\sqsubseteq_{pmust} R_1$.

Example 5.9 *The axiom $P \sqcap (Q \,\Box\, R) = (P \sqcap Q) \,\Box\, (P \sqcap R)$ is unsound.*

Let $R_1 = (a \,{}_{\frac{1}{2}}\!\oplus\, b) \sqcap ((a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\Box\, \mathbf{0})$ and $R_2 = ((a \,{}_{\frac{1}{2}}\!\oplus\, b) \sqcap (a \,{}_{\frac{1}{2}}\!\oplus\, b)) \,\Box\, ((a \,{}_{\frac{1}{2}}\!\oplus\, b) \sqcap \mathbf{0})$.
One obtains $\mathcal{A}pply(a.\omega, R_1) = \{\frac{1}{2}\}$ and $\mathcal{A}pply(a.\omega, R_2) = \{\frac{1}{2}, \frac{3}{4}\}$. So $R_2 \not\sqsubseteq_{pmay} R_1$. Let R_3 and R_4 result from substituting $a \,{}_{\frac{1}{2}}\!\oplus\, b$ for each of P, Q and R in the axiom above. Now $\mathcal{A}pply(a.\omega, R_3) = \{\frac{1}{2}, \frac{3}{4}\}$ and $\mathcal{A}pply(a.\omega, R_4) = \{\frac{3}{4}\}$. So $R_4 \not\sqsubseteq_{pmust} R_3$.

Example 5.10 *The axiom $P \,{}_{p}\!\oplus\, (Q \,\Box\, R) = (P \,{}_{p}\!\oplus\, Q) \,\Box\, (P \,{}_{p}\!\oplus\, R)$ is unsound.*

Let $R_1 = a \,{}_{\frac{1}{2}}\!\oplus\, (b \,\Box\, c)$, $R_2 = (a \,{}_{\frac{1}{2}}\!\oplus\, b) \,\Box\, (a \,{}_{\frac{1}{2}}\!\oplus\, c)$ and $R_3 = (a \,\Box\, b) \,{}_{\frac{1}{2}}\!\oplus\, (a \,\Box\, c)$. R_1 is an instance of the left-hand side of the axiom, and R_2 an instance of the right-hand side. Here we use R_3 as a tool to reason about R_2, but in Section 5.14.4 we need R_3 in its own right. Note that $[\![R_2]\!] = \frac{1}{2} \cdot [\![R_1]\!] + \frac{1}{2} \cdot [\![R_3]\!]$. Let $T = a.\omega$. It is easy to see that $\mathcal{A}pply(T, R_1) = \{\frac{1}{2}\}$ and $\mathcal{A}pply(T, R_3) = \{1\}$. Therefore $\mathcal{A}pply(T, R_2) = \{\frac{3}{4}\}$. So we have $R_2 \not\sqsubseteq_{pmay} R_1$ and $R_2 \not\sqsubseteq_{pmust} R_1$.

Of all the examples in this section, this is the only one for which we can show that $\sqsubseteq_{pmay}$ and $\sqsupseteq_{pmay}$ both fail, i.e. both inequations that can be associated with the axiom are unsound for may *testing. Let $T = a.(\omega \,{}_{\frac{1}{2}}\!\oplus\, \mathbf{0}) \sqcap (b.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.\omega)$. It is not hard to check that $\mathcal{A}pply(T, R_1) = \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$ and $\mathcal{A}pply(T, R_3) = \{\frac{1}{2}\}$. Thus $\mathcal{A}pply(T, R_2) = \{\frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}\}$. Therefore, we have $R_1 \not\sqsubseteq_{pmay} R_2$.*

For future reference, we also observe that $R_1 \not\sqsubseteq_{pmay} R_3$ and $R_3 \not\sqsubseteq_{pmay} R_1$.

5.4 Must versus may testing

On pCSP there are two differences between the preorders $\sqsubseteq_{\text{pmay}}$ and $\sqsubseteq_{\text{pmust}}$:

- Must testing is more discriminating
- The preorders $\sqsubseteq_{\text{pmay}}$ and $\sqsubseteq_{\text{pmust}}$ are oriented in opposite directions.

In this section we substantiate these claims by proving that $P \sqsubseteq_{\text{pmust}} Q$ implies $Q \sqsubseteq_{\text{pmay}} P$, and by providing a counterexample that shows the implication is strict. We are only able to obtain the implication since our language does not feature *divergence*, infinite sequences of τ-actions. It is well known from the non-probabilistic theory of testing [DNH84, Hen88] that in the presence of divergence $\simeq_{\text{may}}$ and $\simeq_{\text{must}}$ are incomparable.

To establish a relationship between must testing and may testing, we define the context $C[\_] = \_ \mid_{\{\omega\}} (\omega \,\Box\, (\nu \sqcap \nu))$ so that for every test T we obtain a new test $C[T]$, by considering ν instead of ω as success action.

Lemma 5.11 For any process P and test T, it holds that

1. if $p \in \mathcal{A}pply(T, P)$ then $(1-p) \in \mathcal{A}pply(C[T], P)$
2. if $p \in \mathcal{A}pply(C[T], P)$ then there exists a $q \in \mathcal{A}pply(T, P)$ such that $1-q \leq p$.

Proof: A state of the form $C[s] \mid_{Act} t$ can always do a τ-move, and never directly a success action ν. The τ-steps that $C[s] \mid_{Act} t$ can do fall into three classes: the resulting distribution is either

- a point distribution $\overline{v}$ with $v \xrightarrow{\nu}$; we call this a *successful* τ-step, because it contributes 1 to the set $\mathbb{V}(C[s] \mid_{Act} t)$
- a point distribution $\overline{u}$ with u a state from which the success action ν is unreachable; we call this an *unsuccessful* τ-step, because it contributes 0 to the set $\mathbb{V}(C[s] \mid_{Act} t)$
- or a distribution of form $C[\Theta] \mid_{Act} \Delta$.

Note that

- $C[s] \mid_{Act} t$ can always do a successful τ-step
- $C[s] \mid_{Act} t$ can do an unsuccessful τ-step iff $s \mid_{Act} t$ can do a ω-step
- and $C[s] \mid_{Act} t \xrightarrow{\tau} C[\Theta] \mid_{Act} \Delta$ iff $s \mid_{Act} t \xrightarrow{\tau} \Theta \mid_{Act} \Delta$.

Using this, both claims follow by a straightforward induction on T and P. □

Proposition 5.12 If $P \sqsubseteq_{\text{pmust}} Q$ then $Q \sqsubseteq_{\text{pmay}} P$.

Proof: Suppose $P \sqsubseteq_{\text{pmust}} Q$. We must show that, for any test T, if $p \in \mathcal{A}pply(T, Q)$ then there exists a $q \in \mathcal{A}pply(T, P)$ such that $p \leq q$. So suppose $p \in \mathcal{A}pply(T, Q)$. By the first clause of Lemma 5.11, we have $(1-p) \in \mathcal{A}pply(C[T], Q)$. Given that $P \sqsubseteq_{\text{pmust}} Q$, there must be an $x \in \mathcal{A}pply(C[T], P)$ such that $x \leq 1-p$. By the second clause of Lemma 5.11, there exists a $q \in \mathcal{A}pply(T, P)$ such that $1-q \leq x$. It follows that $p \leq q$. Therefore $Q \sqsubseteq_{\text{pmay}} P$. □

Example 5.13 *To show that must testing is strictly more discriminating than may testing consider the processes $a \,\Box\, b$ and $a \sqcap b$, and expose them to test $a.\omega$. It is not hard to see that $\mathcal{A}pply(a.\omega, a \,\Box\, b) = \{1\}$, whereas $\mathcal{A}pply(a.\omega, a \sqcap b) = \{0, 1\}$. Since $\min(\mathcal{A}pply(a.\omega, a \,\Box\, b)) = 1$ and $\min(\mathcal{A}pply(a.\omega, a \sqcap b)) = 0$, using Proposition 4.37 we obtain that $(a \,\Box\, b) \not\sqsubseteq_{pmust} (a \sqcap b)$.*

Since $\max(\mathcal{A}pply(a.\omega, a \,\Box\, b)) = \max(\mathcal{A}pply(a.\omega, a \sqcap b)) = 1$, as a may *test, the test $a.\omega$ does not differentiate between $a \,\Box\, b$ and $a \sqcap b$. In fact, we have $(a \sqcap b) \sqsubseteq_{pmay} (a \,\Box\, b)$, and even $(a \,\Box\, b) \simeq_{pmay} (a \sqcap b)$, but this cannot be shown so easily, as we would have to consider all possible tests. In Section 5.5 we will develop a tool to prove statements $P \sqsubseteq_{pmay} Q$, and apply it to derive the equality above (axiom* (**EI**) *in Figure 5.8).*

5.5 Simulation and failure simulation

The examples of Section 5.3 have been all negative, because one can easily demonstrate an inequivalence between two processes by exhibiting a test which distinguishes them in the appropriate manner. A direct application of the definition of the testing preorders is usually unsuitable for establishing positive results, as this involves a universal quantification over all possible tests that can be applied. To give positive results of the form $P \sqsubseteq_{\rm pmay} Q$ (and similarly for $P \sqsubseteq_{\rm pmust} Q$) we need to come up with a preorder $\sqsubseteq_{\rm finer}$ such that $(P \sqsubseteq_{\rm finer} Q) \Rightarrow (P \sqsubseteq_{\rm pmay} Q)$ and statements $P \sqsubseteq_{\rm finer} Q$ can be obtained by exhibiting a single witness.

In this section we report on investigations in this direction, using *simulations* as our witnesses. We confine ourselves to *may* testing, although similar results hold for *must* testing. The definitions are somewhat complicated by the fact that in a pLTS transitions go from states to distributions; consequently if we are to use sequences of transitions, or *weak transitions* $\stackrel{a}{\Longrightarrow}$ which abstract from sequences of internal actions that might precede or follow the a-transition, then we need to generalise transitions so that they go from distributions to distributions. We first develop the mathematical machinery for doing this.

5.5.1 Lifting relations

Let $\mathcal{R} \subseteq S \times \mathcal{D}(S)$ be a relation from states to distributions. We lift it to a relation $\overline{\mathcal{R}} \subseteq \mathcal{D}(S) \times \mathcal{D}(S)$ by letting $\Delta_1 \,\overline{\mathcal{R}}\, \Delta_2$ whenever

(i) $\Delta_1 = \sum_{i\in I} p_i \cdot \overline{s_i}$, where I is a finite index set and $\sum_{i\in I} p_i = 1$

(ii) For each $i \in I$ there is a distribution Φ_i such that $s_i \,\mathcal{R}\, \Phi_i$

(iii) $\Delta_2 = \sum_{i\in I} p_i \cdot \Phi_i$.

An important point here is that in the decomposition (i) of Δ_1 into $\sum_{i\in I} p_i \cdot \overline{s_i}$, the states s_i are *not necessarily distinct*: that is, the decomposition is not in general unique. Thus when establishing the relationship between Δ_1 and Δ_2, a given state s in Δ_1 may play a number of different roles, and this is seen clearly if we apply this definition to the action relations $\stackrel{\alpha}{\longrightarrow} \subseteq S_p \times \mathcal{D}(S_p)$ in the operational semantics of pCSP. We obtain lifted relations between $\mathcal{D}(S_p)$ and $\mathcal{D}(S_p)$, which to ease the notation we write as $\Delta_1 \stackrel{\alpha}{\longrightarrow} \Delta_2$; then, using pCSP terms to represent distributions, a simple instance of a transition between distributions is given by

$$(a.b \,\Box\, a.c) \,{}_{\frac{1}{2}}\!\oplus\, a.d \quad \stackrel{a}{\longrightarrow} \quad b \,{}_{\frac{1}{2}}\!\oplus\, d$$

But we also have

$$(a.b \,\Box\, a.c) \,{}_{\frac{1}{2}}\!\oplus\, a.d \quad \stackrel{a}{\longrightarrow} \quad (b \,{}_{\frac{1}{2}}\!\oplus\, c) \,{}_{\frac{1}{2}}\!\oplus\, d \tag{5.1}$$

because, viewed as a distribution, the term $(a.b \,\Box\, a.c) \,{}_{\frac{1}{2}}\!\oplus\, a.d$ may be re-written as $((a.b \,\Box\, a.c) \,{}_{\frac{1}{2}}\!\oplus\, (a.b \,\Box\, a.c)) \,{}_{\frac{1}{2}}\!\oplus\, a.d$ representing the sum of point distributions

$$\tfrac{1}{4} \cdot \overline{(a.b \,\Box\, a.c)} + \tfrac{1}{4} \cdot \overline{(a.b \,\Box\, a.c)} + \tfrac{1}{2} \cdot \overline{a.d}$$

from which the move (5.1) can easily be derived using the three moves from states

$$a.b \,\Box\, a.c \stackrel{a}{\longrightarrow} \overline{b} \qquad a.b \,\Box\, a.c \stackrel{a}{\longrightarrow} \overline{c} \qquad a.d \stackrel{a}{\longrightarrow} \overline{d}$$

The lifting construction can also be used to define the concept of a *partial* internal move between distributions, one where part of the distribution does an internal move and the remainder remains unchanged. Write $s \stackrel{\hat{\tau}}{\longrightarrow} \Delta$ if either $s \stackrel{\tau}{\longrightarrow} \Delta$ or $\Delta = \overline{s}$. This relation between states and distributions can be lifted to one between distributions and distributions, and again for notational convenience

we use $\Delta_1 \xrightarrow{\hat{\tau}} \Delta_2$ to denote the lifted relation. As an example, again using process terms to denote distributions, we have

$$(a \sqcap b) \,{}_{\frac{1}{2}}\!\oplus\, (a \sqcap c) \quad \xrightarrow{\hat{\tau}} \quad a \,{}_{\frac{1}{2}}\!\oplus\, (a \sqcap b \;\; {}_{\frac{1}{2}}\!\oplus \;\; c)$$

This follows because as a distribution $(a \sqcap b) \,{}_{\frac{1}{2}}\!\oplus\, (a \sqcap c)$ may be written as

$$\tfrac{1}{4} \cdot \overline{(a \sqcap b)} + \tfrac{1}{4} \cdot \overline{(a \sqcap b)} + \tfrac{1}{4} \cdot \overline{(a \sqcap c)} + \tfrac{1}{4} \cdot \overline{(a \sqcap c)}$$

and we have the four moves from states to distributions:

$$(a \sqcap b) \xrightarrow{\hat{\tau}} \overline{a} \qquad (a \sqcap b) \xrightarrow{\hat{\tau}} \overline{(a \sqcap b)}$$
$$(a \sqcap c) \xrightarrow{\hat{\tau}} \overline{a} \qquad (a \sqcap c) \xrightarrow{\hat{\tau}} \overline{c}$$

5.5.2 The simulation preorder

Following tradition it would be natural to define simulations as relations between states in a pLTS [JYL01, SL94]. However, technically it is more convenient to use relations in $S_p \leftrightarrow \mathcal{D}(S_p)$. One reason may be understood through the example in Figure 5.5. Although in Example 5.2 we found that $R_2 \not\sqsubseteq_{\text{pmay}} R_1$, we do have $R_1 \sqsubseteq_{\text{pmay}} R_2$. If we are to relate these processes via a simulation-like relation, then the initial state of R_1 needs to be related to the initial *distribution* of R_2, containing the two states $a.b$ and $a.c$.

Our definition of simulation uses *weak transitions* [Mil89a], which have the standard definitions except that they now apply to distributions, and $\xrightarrow{\hat{\tau}}$ is used instead of $\xrightarrow{\tau}$. This reflects the understanding that if a distribution may perform a sequence of internal moves before or after executing a visible action, different parts of the distribution may perform different numbers of internal actions:

- Let $\Delta_1 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_2$ whenever $\Delta_1 \xrightarrow{\hat{\tau}}{}^* \Delta_2$.
- Similarly $\Delta_1 \stackrel{\hat{a}}{\Longrightarrow} \Delta_2$ denotes $\Delta_1 \xrightarrow{\hat{\tau}}{}^* \xrightarrow{a} \xrightarrow{\hat{\tau}}{}^* \Delta_2$ whenever $a \in Act$.

We write $s \not\xrightarrow{X}$ with $X \subseteq Act$ when $\forall \alpha \in X \cup \{\tau\} : s \not\xrightarrow{\alpha}$, and $\Delta \not\xrightarrow{X}$ when $\forall s \in \lceil \Delta \rceil : s \not\xrightarrow{X}$.

Definition 5.14 *A relation $\mathcal{R} \subseteq S \times \mathcal{D}(S)$ is said to be a* failure simulation *if for all $s, \Theta, \alpha, \Delta$ we have that*

- $s \,\mathcal{R}\, \Theta \wedge s \xrightarrow{\alpha} \Delta$ *implies* $\exists \Theta' : \Theta \stackrel{\hat{\alpha}}{\Longrightarrow} \Theta' \wedge \Delta \,\overline{\mathcal{R}}\, \Theta'$
- $s \,\mathcal{R}\, \Theta \wedge s \not\xrightarrow{X}$ *implies* $\exists \Theta' : \Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta' \wedge \Theta' \not\xrightarrow{X}$.

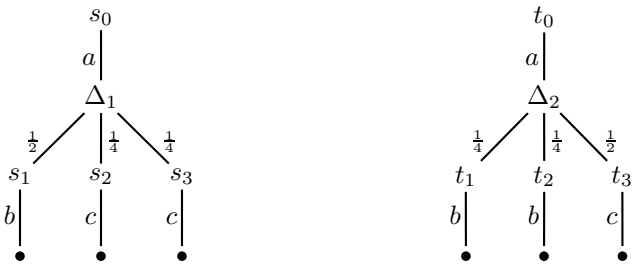
We write $s \lhd_{FS} \Theta$ to mean that there is some failure simulation $\mathcal{R}$ such that $s \,\mathcal{R}\, \Theta$. Similarly, we define simulation *and $s \lhd_S \Theta$ by dropping the second clause in Definition 5.14.*

Definition 5.15 *The* simulation preorder $\sqsubseteq_S$ *and* failure simulation preorder $\sqsubseteq_{FS}$ *on* pCSP *are defined as follows:*

$$P \sqsubseteq_S Q \quad \textit{iff} \quad [\![Q]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Theta \textit{ for some } \Theta \textit{ with } [\![P]\!] \,\overline{\lhd_S}\, \Theta$$
$$P \sqsubseteq_{FS} Q \quad \textit{iff} \quad [\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Theta \textit{ for some } \Theta \textit{ with } [\![Q]\!] \,\overline{\lhd_{FS}}\, \Theta\,.$$

(Note the opposing directions.) The equivalences generated by $\sqsubseteq_S$ and $\sqsubseteq_{FS}$ are called (failure) simulation equivalence*, denoted $\simeq_S$ and $\simeq_{FS}$, respectively.*

If $P \in S_p$, that is if P is a state in the pLTS of pCSP and so $[\![P]\!] = \overline{P}$, then to establish $P \sqsubseteq_S Q$ it is sufficient to exhibit a simulation between the state P and the distribution $[\![Q]\!]$, because trivially $s \lhd_S \Delta$ implies $\overline{s} \,\overline{\lhd_S}\, \Delta$.



$P_1 = a.(b \,{}_{\frac{1}{2}}\!\oplus\, (c \,{}_{\frac{1}{2}}\!\oplus\, c))$ $\qquad$ $P_2 = a.((b \,{}_{\frac{1}{2}}\!\oplus\, b) \,{}_{\frac{1}{2}}\!\oplus\, c)$

Figure 5.7: Two simulation equivalent processes

Example 5.16 *Consider the two processes P_i in Figure 5.7. To show $P_1 \sqsubseteq_S P_2$ it is sufficient to exhibit a simulation $\mathcal{R}$ such that $s_0 \,\mathcal{R}\, \overline{t_0}$. Let $\mathcal{R} \subseteq S_p \times \mathcal{D}(S_p)$ be defined by*

$$s_0 \;\mathcal{R}\; \overline{t_0} \qquad s_1 \;\mathcal{R}\; \Delta_t \qquad s_2 \;\mathcal{R}\; \overline{t_3} \qquad s_3 \;\mathcal{R}\; \overline{t_3} \qquad \mathbf{0} \;\mathcal{R}\; \overline{\mathbf{0}}$$

where Δ_t is the two-point distribution mapping both t_1 and t_2 to the probability $\frac{1}{2}$. Then it is straightforward to check that it satisfies the requirements of a simulation: the only non-trivial requirement is that $\Delta_1 \,\overline{\mathcal{R}}\, \Delta_2$. But this follows from the fact that

$$\begin{array}{rcl} \Delta_1 & = & \frac{1}{2} \cdot \overline{s_1} + \frac{1}{4} \cdot \overline{s_2} + \frac{1}{4} \cdot \overline{s_3} \\ \Delta_2 & = & \frac{1}{2} \cdot \Delta_t + \frac{1}{4} \cdot \overline{t_3} + \frac{1}{4} \cdot \overline{t_3} \end{array}$$

As another example reconsider $R_1 = a.(b \,{}_{\frac{1}{2}}\!\oplus\, c)$ and $R_2 = a.b \,{}_{\frac{1}{2}}\!\oplus\, a.c$ from Figure 5.5, where for convenience we use process terms to denote their semantic interpretations. It is easy to see that $R_1 \sqsubseteq_S R_2$ because of the simulation

$$R_1 \;\mathcal{R}\; [\![R_2]\!] \qquad b \;\mathcal{R}\; \overline{b} \qquad c \;\mathcal{R}\; \overline{c} \qquad \mathbf{0} \;\mathcal{R}\; \overline{\mathbf{0}}$$

Namely $R_2 \xrightarrow{a} (b \,{}_{\frac{1}{2}}\!\oplus\, c)$ and $(b \,{}_{\frac{1}{2}}\!\oplus\, c) \,\overline{\mathcal{R}}\, (b \,{}_{\frac{1}{2}}\!\oplus\, c)$.

Similarly $(a \,{}_{\frac{1}{2}}\!\oplus\, c) \sqcap (b \,{}_{\frac{1}{2}}\!\oplus\, c) \sqsubseteq_S (a \sqcap b) \,{}_{\frac{1}{2}}\!\oplus\, c$ because it is possible to find a simulation between the state $(a \,{}_{\frac{1}{2}}\!\oplus\, c) \sqcap (b \,{}_{\frac{1}{2}}\!\oplus\, c)$ and the distribution $(a \sqcap b) \,{}_{\frac{1}{2}}\!\oplus\, c$.

In case $P \notin S_p$, a statement $P \sqsubseteq_S Q$ cannot always be established by a simulation $\mathcal{R}$ such that $[\![P]\!] \,\overline{\mathcal{R}}\, [\![Q]\!]$.

Example 5.17 *Compare the processes $P = a \,{}_{\frac{1}{2}}\!\oplus\, b$ and $P \sqcap P$. Note that $[\![P]\!]$ is the distribution $\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}$ whereas $[\![P \sqcap P]\!]$ is the point distribution $\overline{P \sqcap P}$. The relation $\mathcal{R}$ given by*

$$(P \sqcap P) \;\mathcal{R}\; (\tfrac{1}{2}\,\overline{a} + \tfrac{1}{2}\,\overline{b}) \qquad a \;\mathcal{R}\; \overline{a} \qquad b \;\mathcal{R}\; \overline{b} \qquad \mathbf{0} \;\mathcal{R}\; \overline{\mathbf{0}}$$

is a simulation, because the τ-step $P \sqcap P \xrightarrow{\tau} (\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b})$ can be matched by the idle transition $(\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}) \stackrel{\hat{\tau}}{\Longrightarrow} (\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b})$, and we have $(\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}) \,\overline{\mathcal{R}}\, (\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b})$. Thus $(P \sqcap P) \lhd_S (\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}) = [\![P]\!]$, hence $[\![P \sqcap P]\!] \,\overline{\lhd_S}\, [\![P]\!]$, and therefore $P \sqcap P \sqsubseteq_S P$.

This type of reasoning does not apply to the other direction. Any simulation $\mathcal{R}$ with $(\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}) \,\overline{\mathcal{R}}\, \overline{P \sqcap P}$ would have to satisfy $a \,\mathcal{R}\, \overline{P \sqcap P}$ and $b \,\mathcal{R}\, \overline{P \sqcap P}$. However, the move $a \xrightarrow{a} \mathbf{0}$ cannot be matched by the process $\overline{P \sqcap P}$, as the only transition the latter process can do is $\overline{P \sqcap P} \xrightarrow{\tau} (\frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b})$, and only half of that distribution can match the a-move. Thus, no such simulation exists, and we find $[\![P]\!] \,\overline{\not\lhd_S}\, [\![P \sqcap P]\!]$. Nevertheless, we still have $P \sqsubseteq_S P \sqcap P$. Here, the transition $\stackrel{\hat{\tau}}{\Longrightarrow}$ from Definition 5.15 comes to the rescue. As $[\![P \sqcap P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![P]\!]$ and $[\![P]\!] \,\overline{\lhd_S}\, [\![P]\!]$, we obtain $P \sqsubseteq_S P \sqcap P$.

Example 5.18 *Let $P = a \,{}_{\frac{1}{2}}\!\oplus\, b$ and $Q = P \,\Box\, P$. We have $P \sqsubseteq_S Q$ because $[\![P]\!] \,\overline{\lhd_S}\, [\![Q]\!]$ which comes from the following observations:*

1. $[\![P]\!] = \frac{1}{2}\,\overline{a} + \frac{1}{2}\,\overline{b}$
2. $[\![Q]\!] = \frac{1}{2}(\frac{1}{2}\,\overline{a \,\Box\, a} + \frac{1}{2}\,\overline{a \,\Box\, b}) + \frac{1}{2}(\frac{1}{2}\,\overline{a \,\Box\, b} + \frac{1}{2}\,\overline{b \,\Box\, b})$
3. $a \lhd_S (\frac{1}{2}\,\overline{a \,\Box\, a} + \frac{1}{2}\,\overline{a \,\Box\, b})$
4. $b \lhd_S (\frac{1}{2}\,\overline{a \,\Box\, b} + \frac{1}{2}\,\overline{b \,\Box\, b})$

This kind of reasoning does not apply to $\lhd_{FS}$. For example, we have $a \not\lhd_{FS} (\frac{1}{2}\,\overline{a \,\Box\, a} + \frac{1}{2}\,\overline{a \,\Box\, b})$ because the state on the left hand side can refuse to do action b while the distribution on the right hand side cannot. Indeed, it holds that $Q \not\sqsubseteq_{FS} P$.

Because of the asymmetric use of distributions in the definition of simulations it is not immediately obvious that $\sqsubseteq_S$ and $\sqsubseteq_{FS}$ are actually preorders (reflexive and transitive relations) and hence $\simeq_S$ and $\simeq_{FS}$ are equivalence relations. In order to show this, we first need to establish some properties of $\lhd_S$ and $\lhd_{FS}$.

Lemma 5.19 Suppose $\sum_{i\in I} p_i = 1$ and $\Delta_i \stackrel{\hat{\alpha}}{\Longrightarrow} \Phi_i$ for each $i \in I$, with I a finite index set. Then

$$\sum_{i\in I} p_i \cdot \Delta_i \stackrel{\hat{\alpha}}{\Longrightarrow} \sum_{i\in I} p_i \cdot \Phi_i$$

Proof: We first prove the case $\alpha = \tau$. For each $i \in I$ there is a number k_i such that $\Delta_i = \Delta_{i0} \stackrel{\hat{\tau}}{\longrightarrow} \Delta_{i1} \stackrel{\hat{\tau}}{\longrightarrow} \Delta_{i2} \stackrel{\hat{\tau}}{\longrightarrow} \cdots \stackrel{\hat{\tau}}{\longrightarrow} \Delta_{ik_i} = \Delta_i'$. Let $k = \max\{k_i \mid i \in I\}$, using that I is finite. Since we have $\Phi \stackrel{\hat{\tau}}{\longrightarrow} \Phi$ for any $\Phi \in \mathcal{D}(S)$, we can add spurious transitions to these sequences, until all k_i equal k. After this preparation the lemma follows by k applications of Proposition 4.6(i), taking $\stackrel{\hat{\tau}}{\longrightarrow}$ for $\mathcal{R}$.

The case $\alpha \in Act$ now follows by one more application of Proposition 4.6(i), this time with $\mathcal{R} = \stackrel{a}{\longrightarrow}$, preceded and followed by an application of the case $\alpha = \tau$. □

Lemma 5.20 Suppose $\Delta\ \overline{\lhd_S}\ \Phi$ and $\Delta \stackrel{\alpha}{\longrightarrow} \Delta'$. Then $\Phi \stackrel{\hat{\alpha}}{\Longrightarrow} \Phi'$ for some Φ' such that $\Delta'\ \overline{\lhd_S}\ \Phi'$.

Proof: Similar to the proof of Lemma 4.8. □

Lemma 5.21 Suppose $\Delta\ \overline{\lhd_S}\ \Phi$ and $\Delta \stackrel{\hat{\alpha}}{\Longrightarrow} \Delta'$. Then $\Phi \stackrel{\hat{\alpha}}{\Longrightarrow} \Phi'$ for some Φ' such that $\Delta'\ \overline{\lhd_S}\ \Phi'$.

Proof: First we consider two claims

(i) If $\Delta\ \overline{\lhd_S}\ \Phi$ and $\Delta \stackrel{\hat{\tau}}{\longrightarrow} \Delta'$, then $\Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Phi'$ for some Φ' such that $\Delta'\ \overline{\lhd_S}\ \Phi'$.

(ii) If $\Delta\ \overline{\lhd_S}\ \Phi$ and $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Delta'$, then $\Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Phi'$ for some Φ' such that $\Delta'\ \overline{\lhd_S}\ \Phi'$.

The proof of claim (i) is similar to the proof of Lemma 5.20. Claim (ii) follows from claim (i) by induction on the length of the derivation of $\stackrel{\hat{\tau}}{\Longrightarrow}$. By combining claim (ii) with Lemma 5.20, we obtain the required result. □

Proposition 5.22 The relation $\overline{\lhd_S}$ is both reflexive and transitive on distributions.

Proof: We leave reflexivity to the reader; it relies on the fact that $s \lhd_S \overline{s}$ for every state s.

For transitivity, let $\mathcal{R} \subseteq S_p \times \mathcal{D}(S_p)$ be given by $s\ \mathcal{R}\ \Phi$ iff $s \lhd_S \Delta\ \overline{\lhd_S}\ \Phi$ for some intermediate distribution Δ. Transitivity follows from the two claims

(i) $\Theta\ \overline{\lhd_S}\ \Delta\ \overline{\lhd_S}\ \Phi$ implies $\Theta\ \overline{\mathcal{R}}\ \Phi$

(ii) $\mathcal{R}$ is a simulation, hence $\overline{\mathcal{R}} \subseteq \overline{\lhd_S}$.

Claim (ii) is a straightforward application of Lemma 5.21, so let us look at (i). From $\Theta \mathrel{\overline{\lhd_S}} \Delta$ we have

$$\Theta = \sum_{i\in I} p_i \cdot \overline{s_i}, \qquad s_i \lhd_S \Delta_i, \qquad \Delta = \sum_{i\in I} p_i \cdot \Delta_i$$

Since $\Delta \mathrel{\overline{\lhd_S}} \Phi$, from part (ii) of Proposition 4.6 we know $\Phi = \sum_{i\in I} p_i \cdot \Phi_i$ such that $\Delta_i \mathrel{\overline{\lhd_S}} \Phi_i$. So for each i we have $s_i \mathcal{R} \Phi_i$, from which it follows that $\Theta \mathrel{\overline{\mathcal{R}}} \Phi$. □

Proposition 5.23 $\sqsubseteq_S$ and $\sqsubseteq_{FS}$ are preorders, i.e. they are reflexive and transitive.

Proof: By combination of Lemma 5.21 and Proposition 5.22, we obtain that $\sqsubseteq_S$ is a preorder. The case for $\sqsubseteq_{FS}$ can be similarly established by proving the counterparts of Lemma 5.21 and Proposition 5.22. □

5.5.3 The simulation preorders are precongruences

In Theorem 5.26 of this section we establish that the pCSP operators are monotone w.r.t. the simulation preorders, i.e. that both $\sqsubseteq_S$ and $\sqsubseteq_{FS}$ are precongruences for pCSP. This implies that the pCSP operators are compositional for them or, equivalently, that $\simeq_S$ and $\simeq_{FS}$are congruences for pCSP. The following two lemmas gather some facts we need in the proof of this theorem. Their proofs are straightforward, although somewhat tedious.

Lemma 5.24 *(i) If* $\Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Phi'$ *then* $\Phi \Box \Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Phi' \Box \Delta$ *and* $\Delta \Box \Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Delta \Box \Phi'$.

(ii) If $\Phi \stackrel{a}{\longrightarrow} \Phi'$ *then* $\Phi \Box \Delta \stackrel{a}{\longrightarrow} \Phi'$ *and* $\Delta \Box \Phi \stackrel{a}{\longrightarrow} \Phi'$.

(iii) $(\sum_{j\in J} p_j \cdot \Phi_j) \Box (\sum_{k\in K} q_k \cdot \Delta_k) \;=\; \sum_{j\in J}\sum_{k\in K} (p_j \cdot q_k) \cdot (\Phi_j \Box \Delta_k)$.

(iv) Given relations $\mathcal{R}, \mathcal{R}' \subseteq S_p \times \mathcal{D}(S_p)$ *satisfying* $s\mathcal{R}'\Delta$ *whenever* $s = s_1 \Box s_2$ *and* $\Delta = \Delta_1 \Box \Delta_2$ *with* $s_1 \mathcal{R} \Delta_1$ *and* $s_2 \mathcal{R} \Delta_2$. *Then* $\Phi_i \mathrel{\overline{\mathcal{R}}} \Delta_i$ *for* $i = 1, 2$ *implies* $(\Phi_1 \Box \Phi_2) \mathrel{\overline{\mathcal{R}'}} (\Delta_1 \Box \Delta_2)$. □

Lemma 5.25 *(i) If* $\Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Phi'$ *then* $\Phi \mid_A \Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Phi' \mid_A \Delta$ *and* $\Delta \mid_A \Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Delta \mid_A \Phi'$.

(ii) If $\Phi \stackrel{a}{\longrightarrow} \Phi'$ *and* $a \notin A$ *then* $\Phi \mid_A \Delta \stackrel{a}{\longrightarrow} \Phi' \mid_A \Delta$ *and* $\Delta \mid_A \Phi \stackrel{a}{\longrightarrow} \Delta \mid_A \Phi'$.

(iii) If $\Phi \stackrel{a}{\longrightarrow} \Phi'$, $\Delta \stackrel{a}{\longrightarrow} \Delta'$ *and* $a \in A$ *then* $\Delta \mid_A \Phi \stackrel{\tau}{\longrightarrow} \Delta' \mid_A \Phi'$.

(iv) $(\sum_{j\in J} p_j \cdot \Phi_j) \mid_A (\sum_{k\in K} q_k \cdot \Delta_k) = \sum_{j\in J}\sum_{k\in K} (p_j \cdot q_k) \cdot (\Phi_j \mid_A \Delta_k)$.

(v) Given relations $\mathcal{R}, \mathcal{R}' \subseteq S_p \times \mathcal{D}(S_p)$ *satisfying* $s\mathcal{R}'\Delta$ *whenever* $s = s_1 \mid_A s_2$ *and* $\Delta = \Delta_1 \mid_A \Delta_2$ *with* $s_1 \mathcal{R} \Delta_1$ *and* $s_2 \mathcal{R} \Delta_2$. *Then* $\Phi_i \mathrel{\overline{\mathcal{R}}} \Delta_i$ *for* $i = 1, 2$ *implies* $(\Phi_1 \mid_A \Phi_2) \mathrel{\overline{\mathcal{R}'}} (\Delta_1 \mid_A \Delta_2)$. □

Theorem 5.26 *Let* $\sqsubseteq \in \{\sqsubseteq_S, \sqsubseteq_{FS}\}$. *Suppose* $P_i \sqsubseteq Q_i$ *for* $i = 1, 2$. *Then*

1. $a.P_1 \sqsubseteq a.Q_1$
2. $P_1 \sqcap P_2 \sqsubseteq Q_1 \sqcap Q_2$
3. $P_1 \Box P_2 \sqsubseteq Q_1 \Box Q_2$
4. $P_1 \,{}_p\!\oplus P_2 \sqsubseteq Q_1 \,{}_p\!\oplus Q_2$
5. $P_1 \mid_A P_2 \sqsubseteq Q_1 \mid_A Q_2$

Proof: We first consider the case for $\sqsubseteq_S$.

1. Since $P_1 \sqsubseteq_S Q_1$, there must be a Δ_1 such that $[\![Q_1]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1$ and $[\![P_1]\!] \mathrel{\overline{\lhd_S}} \Delta_1$. It is easy to see that $a.P_1 \lhd_S \overline{a.Q_1}$ because the transition $a.P_1 \stackrel{a}{\longrightarrow} [\![P_1]\!]$ can be matched by $\overline{a.Q_1} \stackrel{a}{\longrightarrow} [\![Q_1]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1$. Thus $[\![a.P_1]\!] = \overline{a.P_1} \mathrel{\overline{\lhd_S}} \overline{a.Q_1} = [\![a.Q_1]\!]$.

2. Since $P_i \sqsubseteq_S Q_i$, there must be a Δ_i such that $[\![Q_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i$ and $[\![P_i]\!]\ \overline{\lhd_S}\ \Delta_i$. It is easy to see that $P_1 \sqcap P_2 \lhd_S \overline{Q_1 \sqcap Q_2}$ because the transition $P_1 \sqcap P_2 \stackrel{\tau}{\longrightarrow} [\![P_i]\!]$, for $i = 1$ or $i = 2$, can be matched by $\overline{Q_1 \sqcap Q_2} \stackrel{\tau}{\longrightarrow} [\![Q_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i$. Thus $[\![P_1 \sqcap P_2]\!] = \overline{P_1 \sqcap P_2}\ \overline{\lhd_S}\ \overline{Q_1 \sqcap Q_2} = [\![Q_1 \sqcap Q_2]\!]$.

3. Let $\mathcal{R} \subseteq S_p \times \mathcal{D}(S_p)$ be defined by $s\ \mathcal{R}\ \Delta$ iff either $s \lhd_S \Delta$ or $s = s_1 \Box s_2$ and $\Delta = \Delta_1 \Box \Delta_2$ with $s_1 \lhd_S \Delta_1$ and $s_2 \lhd_S \Delta_2$. We show that $\mathcal{R}$ is a simulation.

   Suppose $s_1 \lhd_S \Delta_1$, $s_2 \lhd_S \Delta_2$ and $s_1 \Box s_2 \stackrel{a}{\longrightarrow} \Theta$ with $a \in Act$. Then $s_i \stackrel{a}{\longrightarrow} \Theta$ for $i = 1$ or $i = 2$. Thus $\Delta_i \stackrel{\hat{a}}{\Longrightarrow} \Delta$ for some Δ with $\Theta\ \overline{\lhd_S}\ \Delta$, and hence $\Theta\ \overline{\mathcal{R}}\ \Delta$. By Lemma 5.24 we have $\Delta_1 \Box \Delta_2 \stackrel{\hat{a}}{\Longrightarrow} \Delta$.

   Now suppose $s_1 \lhd_S \Delta_1$, $s_2 \lhd_S \Delta_2$ and $s_1 \Box s_2 \stackrel{\tau}{\longrightarrow} \Theta$. Then $s_1 \stackrel{\tau}{\longrightarrow} \Phi$ and $\Theta = \Phi \Box \overline{s_2}$ or $s_2 \stackrel{\tau}{\longrightarrow} \Phi$ and $\Theta = \overline{s_1} \Box \Phi$. By symmetry we may restrict attention to the first case. Thus $\Delta_1 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta$ for some Δ with $\Phi\ \overline{\lhd_S}\ \Delta$. By Lemma 5.24 we have $(\Phi \Box \overline{s_2})\ \overline{\mathcal{R}}\ (\Delta \Box \Delta_2)$ and $\Delta_1 \Box \Delta_2 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta \Box \Delta_2$.

   The case that $s \lhd_S \Delta$ is trivial, so we have checked that $\mathcal{R}$ is a simulation indeed. Using this, we proceed to show that $P_1 \Box P_2 \sqsubseteq_S Q_1 \Box Q_2$.

   Since $P_i \sqsubseteq_S Q_i$, there must be a Δ_i such that $[\![Q_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i$ and $[\![P_i]\!]\ \overline{\lhd_S}\ \Delta_i$. By Lemma 5.24, we have $[\![P_1 \Box P_2]\!] = ([\![P_1]\!] \Box [\![P_2]\!])\ \overline{\mathcal{R}}\ (\Delta_1 \Box \Delta_2)$. Therefore $[\![P_1 \Box P_2]\!]\ \overline{\lhd_S}\ (\Delta_1 \Box \Delta_2)$. By Lemma 5.24 we also obtain $[\![Q_1 \Box Q_2]\!] = [\![Q_1]\!] \Box [\![Q_2]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1 \Box [\![Q_2]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1 \Box \Delta_2$, so the required result is established.

4. Since $P_i \sqsubseteq_S Q_i$, there must be a Δ_i such that $[\![Q_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i$ and $[\![P_i]\!]\ \overline{\lhd_S}\ \Delta_i$. Thus $[\![Q_1 \,{}_p\!\oplus\, Q_2]\!] = p \cdot [\![Q_1]\!] + (1{-}p) \cdot [\![Q_2]\!] \stackrel{\hat{\tau}}{\Longrightarrow} p \cdot \Delta_1 + (1{-}p) \cdot \Delta_2$ by Lemma 5.19
   and $[\![P_1 \,{}_p\!\oplus\, P_2]\!] = p \cdot [\![P_1]\!] + (1{-}p) \cdot [\![P_2]\!]\ \overline{\lhd_S}\ p \cdot \Delta_1 + (1{-}p) \cdot \Delta_2$ by Proposition 4.6(i). Hence $P_1 \,{}_p\!\oplus\, P_2 \sqsubseteq_S Q_1 \,{}_p\!\oplus\, Q_2$.

5. Let $\mathcal{R} \subseteq S_p \times \mathcal{D}(S_p)$ be defined by $s\ \mathcal{R}\ \Delta$ iff $s = s_1 \mid_A s_2$ and $\Delta = \Delta_1 \mid_A \Delta_2$ with $s_1 \lhd_S \Delta_1$ and $s_2 \lhd_S \Delta_2$. We show that $\mathcal{R}$ is a simulation. There are three cases to consider.

   (a) Suppose $s_1 \lhd_S \Delta_1$, $s_2 \lhd_S \Delta_2$ and $s_1 \mid_A s_2 \stackrel{\alpha}{\longrightarrow} \Theta_1 \mid_A \overline{s_2}$ because of the transition $s_1 \stackrel{\alpha}{\longrightarrow} \Theta_1$ with $\mu \notin A$. Then $\Delta_1 \stackrel{\hat{\mu}}{\Longrightarrow} \Delta_1'$ for some Δ_1' with $\Theta_1\ \overline{\lhd_S}\ \Delta_1'$. By Lemma 5.25 we have $\Delta_1 \mid_A \Delta_2 \stackrel{\hat{\mu}}{\Longrightarrow} \Delta_1' \mid_A \Delta_2$ and also $(\Theta_1 \mid_A \overline{s_2})\ \overline{\mathcal{R}}\ (\Delta_1' \mid_A \Delta_2)$.

   (b) The symmetric case can be similarly analysed.

   (c) Suppose $s_1 \lhd_S \Delta_1$, $s_2 \lhd_S \Delta_2$ and $s_1 \mid_A s_2 \stackrel{\tau}{\longrightarrow} \Theta_1 \mid_A \Theta_2$ because of the transitions $s_1 \stackrel{a}{\longrightarrow} \Theta_1$ and $s_2 \stackrel{a}{\longrightarrow} \Theta_2$ with $a \in A$. Then for $i = 1$ and $i = 2$ we have $\Delta_i \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i' \stackrel{a}{\longrightarrow} \Delta_i'' \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i'''$ for some $\Delta_i', \Delta_i'', \Delta_i'''$ with $\Theta_1\ \overline{\lhd_S}\ \Delta_i'''$. By Lemma 5.25 we have $\Delta_1 \mid_A \Delta_2 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1' \mid_A \Delta_2' \stackrel{\tau}{\longrightarrow} \Delta_1'' \mid_A \Delta_2'' \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1''' \mid_A \Delta_2'''$ and $(\Theta_1 \mid_A \Theta_2)\ \overline{\mathcal{R}}\ (\Delta_1''' \mid_A \Delta_2''')$.

   So we have checked that $\mathcal{R}$ is a simulation.

   Since $P_i \sqsubseteq_S Q_i$, there must be a Δ_i such that $[\![Q_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i$ and $[\![P_i]\!]\ \overline{\lhd_S}\ \Delta_i$. By Lemma 5.25 we have $[\![P_1 \mid_A P_2]\!] = ([\![P_1]\!] \mid_A [\![P_2]\!])\ \overline{\mathcal{R}}\ (\Delta_1 \mid_A \Delta_2)$. Therefore $[\![P_1 \mid_A P_2]\!]\ \overline{\lhd_S}\ (\Delta_1 \mid_A \Delta_2)$. By Lemma 5.25 we also obtain $[\![Q_1 \mid_A Q_2]\!] = [\![Q_1]\!] \mid_A [\![Q_2]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1 \mid_A [\![Q_2]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1 \mid_A \Delta_2$, which had to be established.

The case for $\sqsubseteq_{FS}$ is analogous. As an example, we show that $\sqsubseteq_{FS}$ is preserved under parallel composition. The key step is to show that the binary relation $\mathcal{R} \subseteq \mathsf{sCSP} \times \mathcal{D}(\mathsf{sCSP})$ defined by

$$\mathcal{R} \ := \ \{(s_1|_A s_2, \Delta_1|_A \Delta_2) \mid s_1 \lhd_{FS} \Delta_1 \wedge s_2 \lhd_{FS} \Delta_2\}.$$

is a failure simulation.

Suppose $s_i \lhd_{FS} \Delta_i$ for $i = 1, 2$ and $s_1 \mid_A s_2 \stackrel{X}{\not\longrightarrow}$ for some $X \subseteq Act$. For each $a \in X$ there are two possibilities:

- If $a \notin A$ then $s_1 \stackrel{a}{\not\longrightarrow}$ and $s_2 \stackrel{a}{\not\longrightarrow}$, since otherwise we would have $s_1 \mid_A s_2 \stackrel{a}{\longrightarrow}$.

- If $a \in A$ then either $s_1 \stackrel{a}{\not\longrightarrow}$ or $s_2 \stackrel{a}{\not\longrightarrow}$, since otherwise we would have $s_1 \mid_A s_2 \stackrel{\tau}{\longrightarrow}$.

Hence we can partition the set X into three subsets: X_0, X_1 and X_2 such that $X_0 = X \backslash A$ and $X_1 \cup X_2 \subseteq A$ with $s_1 \stackrel{X_1}{\not\longrightarrow}$ and $s_2 \stackrel{X_2}{\not\longrightarrow}$, but allowing $s_1 \stackrel{a}{\not\longrightarrow}$ for some $a \in X_2$ and $s_2 \stackrel{a}{\not\longrightarrow}$ for some $a \in X_1$. We then have that $s_i \stackrel{X_0 \cup X_i}{\not\longrightarrow}$ for $i = 1, 2$. By the assumption that $s_i \lhd_{FS} \Delta_i$ for $i = 1, 2$, there is a Δ_i' with $\Delta_i \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_i'$ $\stackrel{X_0 \cup X_i}{\not\longrightarrow}$. Therefore $\Delta_1' |_A \Delta_2' \stackrel{X}{\not\longrightarrow}$ as well. It is stated in [DvGH$^+$07a, Lemma 6.12(i)] that if $\Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Phi'$ then $\Phi \mid_A \Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Phi' \mid_A \Delta$ and $\Delta \mid_A \Phi \stackrel{\hat{\tau}}{\Longrightarrow} \Delta \mid_A \Phi'$. So we have $\Delta_1 \mid_A \Delta_2 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_1' \mid_A \Delta_2'$. Hence $\Delta_1 \mid_A \Delta_2$ can match up the failures of $s_1 \mid_A s_2$.

The matching up of transitions and the using of $\mathcal{R}$ to prove the preservation property of $\sqsubseteq_{FS}$ under parallel composition are similar to those in the corresponding proof for simulations [DvGH$^+$07a, Theorem 6.13(v)], so we omit them. □

5.5.4 Simulations are sound for testing preorders

This section is devoted to the proof that (i) $P \sqsubseteq_S Q$ implies $P \sqsubseteq_{\text{pmay}} Q$ and (ii) $P \sqsubseteq_{FS} Q$ implies $P \sqsubseteq_{\text{pmust}} Q$.

Theorem 5.27 *If $P \sqsubseteq_S Q$ then $P \sqsubseteq_{pmay} Q$.*

Proof: For any test $T \in \mathsf{pCSP}^\omega$ and process $P \in \mathsf{pCSP}$ the set $\mathbb{V}(T \mid_{Act} P)$ is finite, so

$$P \sqsubseteq_{\text{pmay}} Q \text{ iff } \max(\mathbb{V}([\![T \mid_{Act} P]\!])) \leq \max(\mathbb{V}([\![T \mid_{Act} Q]\!])) \text{ for every test } T. \tag{5.2}$$

The following properties for $\Delta_1, \Delta_2 \in \mathsf{pCSP}^\omega$ and $\alpha \in Act_\tau$ are not hard to establish:

$$\Delta_1 \stackrel{\hat{\alpha}}{\Longrightarrow} \Delta_2 \text{ implies } \max(\mathbb{V}(\Delta_1)) \geq \max(\mathbb{V}(\Delta_2)). \tag{5.3}$$

$$\Delta_1 \overline{\lhd_S} \Delta_2 \text{ implies } \max(\mathbb{V}(\Delta_1)) \leq \max(\mathbb{V}(\Delta_2)). \tag{5.4}$$

In [DvGH$^+$07a, Lemma 6.15 and Proposition 6.16] similar properties are proven using a function *maxlive* instead of max$\circ$ $\mathbb{V}$. The same arguments apply here.

Now suppose $P \sqsubseteq_S Q$. Since $\sqsubseteq_S$ is preserved by the parallel operator we have that $T \mid_{Act} P \sqsubseteq_S T \mid_{Act} Q$ for an arbitrary test T. By definition, this means that there is a distribution Δ such that $[\![T \mid_{Act} Q]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta$ and $[\![T \mid_{Act} P]\!] \overline{\lhd_S} \Delta$. By (5.3) and (5.4) we infer that $\max(\mathbb{V}([\![T \mid_{Act} P]\!])) \leq \max(\mathbb{V}([\![T \mid_{Act} Q]\!]))$. The result now follows from (5.2). □

It is tempting to use the same idea to prove that $\sqsubseteq_{FS}$ implies $\sqsubseteq_{\text{pmust}}$, but now using the function min$\circ$ $\mathbb{V}$. However, the min-analogue of Property (5.3) is in general invalid. For example, let R be the process $a \mid_{Act} (a \,\Box\, \omega)$. We have $\min(\mathbb{V}(R)) = 1$, yet $R \stackrel{\tau}{\longrightarrow} \overline{\mathbf{0} \mid_{Act} \mathbf{0}}$ and $\min(\mathbb{V}(\overline{\mathbf{0} \mid_{Act} \mathbf{0}})) = 0$. Therefore, it is not the case that $\Delta_1 \stackrel{\hat{\tau}}{\Longrightarrow} \Delta_2$ implies $\min(\mathbb{V}(\Delta_1)) \leq \min(\mathbb{V}(\Delta_2))$.

Our strategy is therefore as follows. Write $s \stackrel{\alpha}{\longrightarrow}_\omega \Delta$ if both $s \stackrel{\omega}{\not\longrightarrow}$ and $s \stackrel{\alpha}{\longrightarrow} \Delta$ hold. We define $\stackrel{\hat{\tau}}{\longrightarrow}_\omega$ as $\stackrel{\hat{\tau}}{\longrightarrow}$ using $\stackrel{\tau}{\longrightarrow}_\omega$ in place of $\stackrel{\tau}{\longrightarrow}$. Similarly we define $\Longrightarrow_\omega$ and $\stackrel{\hat{\alpha}}{\Longrightarrow}_\omega$. Thus the subscript ω on a transition of any kind indicates that no state is passed through in which ω is enabled. A version of failure simulation adapted to these transition relations is then defined as follows.

Definition 5.28 *Let $\lhd^e_{FS} \subseteq \mathsf{sCSP}^\omega \times \mathcal{D}(\mathsf{sCSP}^\omega)$ be the largest relation such that $s \lhd^e_{FS} \Theta$ implies*

- *if $s \stackrel{\alpha}{\longrightarrow}_\omega \Delta$ then there is some Θ' with $\Theta \stackrel{\hat{\alpha}}{\Longrightarrow}_\omega \Theta'$ and $\Delta \overline{\lhd^e_{FS}} \Theta'$*
- *if $s \stackrel{X}{\not\longrightarrow}$ with $\omega \in X$ then there is some Θ' with $\Theta \stackrel{\hat{\tau}}{\Longrightarrow}_\omega \Theta'$ and $\Theta' \stackrel{X}{\longrightarrow}$.*

Let $P \sqsubseteq^e_{FS} Q$ iff $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow}_\omega \Theta$ for some Θ with $[\![Q]\!] \overline{\lhd^e_{FS}} \Theta$.

Note that for processes P, Q in pCSP (as opposed to pCSP^ω), we have $P \sqsubseteq_{FS} Q$ iff $P \sqsubseteq^e_{FS} Q$.

Proposition 5.29 *If P, Q are processes in pCSP with $P \sqsubseteq_{FS} Q$ and T is a process in pCSP^ω then $T \mid_{Act} P \sqsubseteq^e_{FS} T \mid_{Act} Q$.*

Proof: Similar to the proof of Theorem 5.26. □

Proposition 5.30 *The following properties hold for* $\min\circ\,\mathbb{V}$*, with* $\Delta_1, \Delta_2 \in \mathcal{D}(\mathsf{sCSP}^\omega)$*:*

$$P \sqsubseteq_{pmust} Q \text{ iff } \min(\mathbb{V}([\![T \mid_{Act} P]\!])) \leq \min(\mathbb{V}([\![T \mid_{Act} Q]\!])) \text{ for every test } T. \tag{5.5}$$

$$\Delta_1 \stackrel{\hat{\alpha}}{\Longrightarrow}_\omega \Delta_2 \text{ for } \alpha \in Act_\tau \text{ implies } \min(\mathbb{V}(\Delta_1)) \leq \min(\mathbb{V}(\Delta_2)). \tag{5.6}$$

$$\Delta_1 \overline{\lhd^e_{FS}} \Delta_2 \text{ implies } \min(\mathbb{V}(\Delta_1)) \geq \min(\mathbb{V}(\Delta_2)). \tag{5.7}$$

Proof: Property (5.5) is again straightforward, and Property (5.6) can be established just as in Lemma 6.15 in [DvGH$^+$07a], but with all $\leq$-signs reversed. Property (5.7) follows by structural induction, simultaneously with the property, for $s \in \mathsf{sCSP}^\omega$ and $\Delta \in \mathcal{D}(\mathsf{sCSP}^\omega)$, that

$$s \lhd^e_{FS} \Delta \;\text{ implies }\; \min(\mathbb{V}(s)) \geq \min(\mathbb{V}(\Delta))\,. \tag{5.8}$$

The reduction of Property (5.7) to (5.8) proceeds exactly as in [DvGH$^+$07a, Lemma 6.16(ii)]. For (5.8) itself we distinguish three cases:

- If $s \stackrel{\omega}{\longrightarrow}$, then $\min(\mathbb{V}(s)) = 1 \geq \min(\mathbb{V}(\Delta))$ trivially.
- If $s \stackrel{\omega}{\not\longrightarrow}$ but $s \rightarrow$, then we can closely follow the proof of [DvGH$^+$07a, Lemma 6.16(i)]: Whenever $s \stackrel{\alpha}{\longrightarrow}_\omega \Theta$, for $\alpha \in Act_\tau$ and $\Theta \in \mathcal{D}(\mathsf{sCSP}^\omega)$, then $s \lhd^e_{FS} \Delta$ implies the existence of some Δ_Θ such that $\Delta \stackrel{\hat{\alpha}}{\longrightarrow}^*_\omega \Delta_\Theta$ and $\Theta \overline{\lhd^e_{FS}} \Delta_\Theta$. By induction, using (5.7), it follows that $\min(\mathbb{V}(\Theta)) \geq \min(\mathbb{V}(\Delta_\Theta))$. Consequently, we have that

$$\begin{array}{rcll} \min(\mathbb{V}(s)) & = & \min(\{\min(\mathbb{V}(\Theta)) \mid s \stackrel{\alpha}{\longrightarrow} \Theta\}) & \\ & \geq & \min(\{\min(\mathbb{V}(\Delta_\Theta)) \mid s \stackrel{\alpha}{\longrightarrow} \Theta\}) & \\ & \geq & \min(\{\min(\mathbb{V}(\Delta)) \mid s \stackrel{\alpha}{\longrightarrow} \Theta\}) & \text{(by (5.6))} \\ & = & \min(\mathbb{V}(\Delta))\,. & \end{array}$$

- If $s \not\rightarrow$, that is $s \stackrel{Act^\omega}{\not\longrightarrow}$, then there is some Δ' such that $\Delta \stackrel{\hat{\tau}}{\Longrightarrow}_\omega \Delta'$ and $\Delta' \stackrel{Act^\omega}{\not\longrightarrow}$. By the definition of $\mathbb{V}$, $\min(\mathbb{V}(\Delta'))=0$. Using (5.6), we have $\min(\mathbb{V}(\Delta)) \leq \min(\mathbb{V}(\Delta'))$, so $\min(\mathbb{V}(\Delta))=0$ as well. Thus, also in this case $\min(\mathbb{V}(s)) \geq \min(\mathbb{V}(\Delta))$.

□

Theorem 5.31 *If* $P \sqsubseteq_{FS} Q$ *then* $P \sqsubseteq_{pmust} Q$*.*

Proof: Similar to the proof of Theorem 5.27, using (5.5)–(5.7). □

The next four sections are devoted to proving the converse of Theorems 5.27 and 5.31.

5.6 State- versus action-based testing

Much work on testing [DNH84, YL92, DvGH$^+$07a] uses success *states* marked by outgoing ω-actions; this is referred to as *state-based* testing, which we have used in Section 5.2.5 to define the preorders $\sqsubseteq_{\text{may}}$ and $\sqsubseteq_{\text{must}}$. In other work [Seg96, DvGMZ07], however, it is the *actual execution* of ω that constitutes success. This *action-based* approach is formalised as in the state-based approach, via a modified results-gathering function:

$$\widehat{\mathbb{V}}(s) := \begin{cases} \bigcup\{\widehat{\mathbb{V}}(\Delta) \mid s \stackrel{\alpha}{\longrightarrow} \Delta \wedge \alpha \neq \omega\} \cup \{1 \mid s \stackrel{\omega}{\longrightarrow}\} & \text{if } s \rightarrow \\ \{0\} & \text{otherwise} \end{cases}$$

As in the original $\mathbb{V}$, the α's are non-success actions, including τ; and again, this is done for generality, since in testing outcomes the only non-success action is τ.

If we use this results-gathering function rather than $\mathbb{V}$ in Definition 5.1 we obtain the two slightly different testing preorders, $\widehat{\sqsubseteq}_{\text{pmay}}$ and $\widehat{\sqsubseteq}_{\text{pmust}}$. The following proposition shows that state-based testing is at least as discriminating as action-based testing:

Proposition 5.32

1. *If* $P \sqsubseteq_{pmay} Q$ *then* $P \,\widehat{\sqsubseteq}_{pmay}\, Q$.
2. *If* $P \sqsubseteq_{pmust} Q$ *then* $P \,\widehat{\sqsubseteq}_{pmust}\, Q$.

Proof: For any action-based test $\widehat{T}$ we construct a state-based test T by replacing each subterm $\omega.Q$ by $\tau.\omega$; then we have $\mathbb{V}[\![T \mid_{Act} P]\!] = \widehat{\mathbb{V}}[\![\widehat{T} \mid_{Act} P]\!]$ for all pCSP processes P. □

Proposition 5.32 enables us to reduce our main goal, the converse of Theorems 5.27 and 5.31, to the following property.

Theorem 5.33

1. *If* $P \,\widehat{\sqsubseteq}_{pmay}\, Q$ *then* $P \sqsubseteq_S Q$.
2. *If* $P \,\widehat{\sqsubseteq}_{pmust}\, Q$ *then* $P \sqsubseteq_{FS} Q$.

We set the proof of this theorem as our goal in the next three sections.

Once we have obtained this theorem, it follows that in our framework of finite probabilistic processes the state-based and action-based testing preorders coincide. This result no longer holds in the presence of divergence, at least for must-testing.

Example 5.34 *Suppose we extend our syntax with a state-based process* Ω*, to model divergence, and the operational semantics of Figure 5.1 with the rule*

$$\Omega \xrightarrow{\tau} \overline{\Omega}.$$

It is possible to extend the results-gathering functions $\mathbb{V}$ *and* $\widehat{\mathbb{V}}$ *to these infinite processes, although the definitions are no longer inductive (cf. Definition 5 of [DvGMZ07] or Definition 5.45 of the appendix). In this extended setting we will have* $a.\Omega \not\sqsubseteq_{pmust} a.\Omega \sqcap 0$ *because of the test* $a.\omega$*:*

$$\mathbb{V}([\![a.\omega \mid_{Act} a.\Omega]\!]) = \{1\} \quad \textit{while} \quad \mathbb{V}([\![a.\omega \mid_{Act} a.\Omega \sqcap 0]\!]) = \{0,1\}.$$

This intuitively is due to the fact that the Ω*-encoded divergence of the left-hand process occurs only* after *the first action* a*; and since the left-hand process cannot deadlock before that action, relation* $\sqsubseteq_{must}$ *would prevent the right-hand process from doing so.*

However, a peculiarity of action-based testing is that success actions can be indefinitely inhibited by infinite τ*-branches. We have*

$$\widehat{\mathbb{V}}([\![a.\omega \mid_{Act} a.\Omega]\!]) = \widehat{\mathbb{V}}([\![a.\omega \mid_{Act} a.\Omega \sqcap 0]\!]) = \{0,1\}.$$

Indeed no test can be found to distinguish them, and so one can show $a.\Omega \,\widehat{\sqsubseteq}_{pmust}\, a.\Omega \sqcap 0$.

Note that probabilistic behaviour plays no role in this counter-example. In CSP (without probabilities) there is no difference between $\widehat{\sqsubseteq}_{\text{may}}$ and $\sqsubseteq_{\text{may}}$, whereas $\widehat{\sqsubseteq}_{\text{must}}$ is strictly less discriminating than $\sqsubseteq_{\text{must}}$. For finitely branching processes, the CSP refinement preorder based on failures and divergences [BHR84, Hoa85b, OH86] coincides with the state-based relation $\sqsubseteq_{\text{must}}$.

5.7 Vector-based testing

This section describes another variation on testing, a richer testing framework due to Segala [Seg96], in which countably many success actions exist: the application of a test to a process yields a set of *vectors* over the real numbers, rather than a set of scalars. The resulting action-based testing preorders will serve as a stepping stone in proving Theorem 5.33.

Let Ω be a *set* of fresh success actions with $\Omega \cap Act_\tau = \emptyset$. An Ω-test is again a pCSP process, but this time allowing subterms $\omega.P$ for any $\omega \in \Omega$. Applying such a test to a process yields a non-empty set of test outcome-*tuples* $\widehat{\mathcal{A}}^{\Omega}(T,P) \subseteq [0,1]^{\Omega}$. As with standard scalar testing, each outcome arises

from a resolution of the nondeterministic choices in $T \mid_{Act} P$. However, here an outcome is a *tuple* and its ω-component gives the probability that this resolution will perform the success action ω.

For vector-based testing we again inductively define a results-gathering function, but first we require some auxiliary notation. For any action α define $\alpha! : [0,1]^\Omega \to [0,1]^\Omega$ by

$$\alpha! o(\omega) = \begin{cases} 1 & \text{if } \omega = \alpha \\ o(\omega) & \text{otherwise} \end{cases}$$

so that if α is a success action, in Ω, then $\alpha!$ updates the tuple to 1 at that point, leaving it unchanged otherwise, and when $\alpha \notin \Omega$ the function $\alpha!$ is the identity. These functions lift to sets $O \subseteq [0,1]^\Omega$ as usual, via $\alpha! O := \{\alpha! o \mid o \in O\}$.

Next, for any set X define its *convex closure* $\updownarrow X$ by

$$\updownarrow X := \{\, \textstyle\sum_{i\in I} p_i o_i \mid p \in \mathcal{D}(I) \text{ and } o : I \to X \,\} .$$

Here, as usual, I is assumed to be a finite index set. Finally, $\vec{0} \in [0,1]^\Omega$ is given by $\vec{0}(\omega) = 0$ for all $\omega \in \Omega$. Let pCSP^Ω be the set of Ω-tests, and sCSP^Ω the set of state-based Ω-tests.

Definition 5.35 *The* action-based, vector-based, convex-closed results-gathering function $\widehat{\mathbb{V}}^\Omega_\updownarrow : \mathsf{sCSP}^\Omega \to \mathcal{P}([0,1]^\Omega)$ *is given by*

$$\widehat{\mathbb{V}}^\Omega_\updownarrow(s) \;:=\; \begin{cases} \updownarrow \bigcup \{\, \alpha!(\widehat{\mathbb{V}}^\Omega_\updownarrow(\Delta)) \mid s \xrightarrow{\alpha} \Delta,\ \alpha \in \Omega \cup Act_\tau \,\} & \textit{if } s \to \\ \{\vec{0}\} & \textit{otherwise} \end{cases} \tag{5.9}$$

As with our previous results-gathering functions $\mathbb{V}$ and $\widehat{\mathbb{V}}$, this function extends to the type $\mathcal{D}(\mathsf{sCSP}^\Omega) \to \mathcal{P}([0,1]^\Omega)$ via the convention $\widehat{\mathbb{V}}^\Omega_\updownarrow(\Delta) := Exp_\Delta \widehat{\mathbb{V}}^\Omega_\updownarrow$.

For any pCSP *process P and Ω-test T, let*

$$\widehat{\mathcal{A}}^\Omega_\updownarrow(T,P) \;:=\; \widehat{\mathbb{V}}^\Omega_\updownarrow[\![T \mid_{Act} P]\!] .$$

The vector-based may- *and* must *preorders are given by*

$$P \;\widehat{\sqsubseteq}^\Omega_{pmay}\; Q \;\text{ iff }\; \textit{for all } \Omega\textit{-tests } T\textit{: } \widehat{\mathcal{A}}^\Omega_\updownarrow(T,P) \sqsubseteq_{\mathrm{Ho}} \widehat{\mathcal{A}}^\Omega_\updownarrow(T,Q)$$
$$P \;\widehat{\sqsubseteq}^\Omega_{pmust}\; Q \;\text{ iff }\; \textit{for all } \Omega\textit{-tests } T\textit{: } \widehat{\mathcal{A}}^\Omega_\updownarrow(T,P) \sqsubseteq_{\mathrm{Sm}} \widehat{\mathcal{A}}^\Omega_\updownarrow(T,Q)$$

where $\sqsubseteq_{\mathrm{Ho}}$ and $\sqsubseteq_{\mathrm{Sm}}$ are the Hoare- and Smyth preorders on $\mathcal{P}([0,1]^\Omega)$ generated from $\leq$ index-wise on $[0,1]^\Omega$ itself.

We will explain the rôle of convex-closure $\updownarrow$ in this definition. Let $\widehat{\mathbb{V}}^\Omega_\updownarrow$ be defined as $\widehat{\mathbb{V}}^\Omega_\updownarrow$ above, but omitting the use of $\updownarrow$. It is easy to see that $\widehat{\mathbb{V}}^\Omega_\updownarrow(s) = \updownarrow \widehat{\mathbb{V}}^\Omega_\updownarrow(s)$ for all $s \in \mathsf{sCSP}^\Omega$.

Applying convex closure to subsets of the one-dimensional interval $[0,1]$ (such as arise from applying scalar tests to processes) has no effect on the Hoare and Smyth orders between these subsets:

Lemma 5.36 *Suppose $X,\ Y \subseteq [0,1]$. Then*

1. *$X \sqsubseteq_{\mathrm{Ho}} Y$ if and only if $\updownarrow X \sqsubseteq_{\mathrm{Ho}} \updownarrow Y$.*
2. *$X \sqsubseteq_{\mathrm{Sm}} Y$ if and only if $\updownarrow X \sqsubseteq_{\mathrm{Sm}} \updownarrow Y$.*

Proof: We restrict attention to (1); the proof of (2) goes likewise. It suffices to show that (i) $X \sqsubseteq_{\mathrm{Ho}} \updownarrow X$ and (ii) $\updownarrow X \sqsubseteq_{\mathrm{Ho}} X$. We only prove (ii) since (i) is obvious. Suppose $x \in \updownarrow X$, then $x = \sum_{i\in I} p_i x_i$ for a finite set I with $\sum_{i\in I} p_i = 1$ and $x_i \in X$. Let $x^* = \max\{x_i \mid i \in I\}$. Then

$$x \;=\; \sum_{i\in I} p_i x_i \;\leq\; \sum_{i\in I} p_i x^* \;=\; x^* \;\in\; X.$$

□

It follows that for scalar testing it makes no difference whether convex closure is employed or not. Vector-based testing, as proposed in Definition 5.35, is a conservative extension of action-based testing, as described in Section 5.6:

Corollary 5.37 *Suppose Ω is the singleton set $\{\omega\}$. Then*

1. *$P \widehat{\sqsubseteq}^{\Omega}_{pmay} Q$ if and only if $P \widehat{\sqsubseteq}_{pmay} Q$.*
2. *$P \widehat{\sqsubseteq}^{\Omega}_{pmust} Q$ if and only if $P \widehat{\sqsubseteq}_{pmust} Q$.*

Proof: $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow} = \updownarrow\widehat{\mathbb{V}}^{\Omega}_{\updownarrow} = \updownarrow\widehat{\mathbb{V}}$ when Ω is $\{\omega\}$, so the result follows from Lemma 5.36. □

Lemma 5.36 does not generalise to $[0,1]^k$, when $k > 1$, as the following example demonstrates:

Example 5.38 *Let X, Y denote $\{(0.5, 0.5)\}$, $\{(1,0),(0,1)\}$ respectively. Then it is easy to show that $\updownarrow X \sqsubseteq_{\mathrm{Ho}} \updownarrow Y$ although obviously $X \not\sqsubseteq_{\mathrm{Ho}} Y$.*

This example can be exploited to show that for vector-based testing it *does* make a difference whether convex closure is employed.

Example 5.39 *Consider the two processes*

$$P := a \,{}_{\frac{1}{2}}\!\oplus\, b \quad \text{and} \quad Q := a \sqcap b \ .$$

Take $\Omega = \{\omega_1, \omega_2\}$. Employing the results-gathering function $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}$, without convex closure, with the test $T := a.\omega_1 \,\square\, b.\omega_2$ we obtain

$$\begin{array}{lcl} \widehat{\mathcal{A}}^{\Omega}(T,P) & = & \{(0.5,0.5)\} \\ \widehat{\mathcal{A}}^{\Omega}(T,Q) & = & \{(1,0),(0,1)\} \ . \end{array}$$

As pointed out in Example 5.38, this entails $\widehat{\mathcal{A}}^{\Omega}(T,P) \not\sqsubseteq_{\mathrm{Ho}} \widehat{\mathcal{A}}^{\Omega}(T,Q)$, although their convex closures $\widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T,P)$ and $\widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T,Q)$ are *related under the Hoare preorder.*

Convex closure is a uniform way of ensuring that internal choice can simulate an arbitrary probabilistic choice [HSM97]. For the processes P and Q of Example 5.39 it is obvious that $P \sqsubseteq_S Q$, and from Theorem 5.27 it therefore follows that $P \sqsubseteq_{\mathrm{pmay}} Q$. This fits with the intuition that a probabilistic choice is an acceptable implementation of a nondeterministic choice occurring in a specification. Considering that we use $\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmay}}$ as a stepping stone in showing the coincidence of $\sqsubseteq_S$ and $\sqsubseteq_{\mathrm{pmay}}$, we must have $P \widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmay}} Q$. For this reason we use convex closure in Definition 5.35.

In [DvGMZ07] the results-gathering function $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}$ with $\Omega = \{\omega_1, \omega_2, \cdots\}$ was called simply $\mathbb{W}$ (because action-based/vector-based/convex-closed testing was assumed there throughout, making the $\widehat{\cdot}^{\Omega}_{\updownarrow}$-indicators superfluous); and it was defined in terms of a formalisation of the notion of a resolution. As we show in Proposition 5.48 of the appendix, the inductive Definition 5.35 above yields the same results. In the present paper our interest in vector-based testing stems from the following result.

Theorem 5.40

1. *$P \widehat{\sqsubseteq}^{\Omega}_{pmay} Q$ iff $P \widehat{\sqsubseteq}_{pmay} Q$*
2. *$P \widehat{\sqsubseteq}^{\Omega}_{pmust} Q$ iff $P \widehat{\sqsubseteq}_{pmust} Q$.* □

Proof: In [DvGMZ07, Theorem 3] this theorem has been established for versions of $\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmay}}$ and $\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmust}}$ where tests are finite probabilistic automata, as defined in Section 5.8. The key argument is that when $P \widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmay}} Q$ can be refuted by means of a vector-based test T, then $P \widehat{\sqsubseteq}_{\mathrm{pmay}} Q$ can be refuted by means of a scalar test $T \| U$, where U is administrative code which collates the vector of results produced by T and effectively renders them as a unique scalar result, and similarly for $\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmust}}$. This theorem applies to our setting as well, due to the observation that if a test T can be represented as a pCSP^{Ω}-expression, then so can the test $T \| U$. □

Because of Theorem 5.40, in order to establish Theorem 5.33 it will suffice to show that

- $P \,\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmay}}\, Q$ implies $P \sqsubseteq_S Q$ and
- $P \,\widehat{\sqsubseteq}^{\Omega}_{\mathrm{pmust}}\, Q$ implies $P \sqsubseteq_{FS} Q$.

This shift from scalar testing to vector-based testing is motivated by the fact that the latter enables us to use more informative tests, allowing us to discover more intensional properties of the processes being tested.

The crucial characteristics of $\widehat{\mathcal{A}}^{\Omega}_{\downarrow}$ needed for the above implications are summarised in Lemmas 5.41 and 5.42. For convenience of presentation, we write $\vec{\omega}$ for the vector in $[0,1]^{\Omega}$ defined by $\vec{\omega}(\omega) = 1$ and $\vec{\omega}(\omega') = 0$ for $\omega' \neq \omega$. Sometimes we treat a distribution Δ of finite support as the pCSP expression $\bigoplus_{s \in \lceil \Delta \rceil} \Delta(s) \cdot s$, so that $\widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, \Delta) := \mathrm{Exp}_{\Delta} \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, \_)$.

Lemma 5.41 *Let P be a* pCSP *process, and T, T_i be tests.*

1. $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\omega, P)$ *iff* $o = \vec{\omega}$.
2. $\vec{0} \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\square_{a \in X} a.\omega, P)$ *iff* $\exists \Delta : [\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Delta \not\xrightarrow{X}$.
3. *Suppose the action ω does not occur in the test T. Then $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\omega \,\square\, a.T, P)$ with $o(\omega) = 0$ iff there is a $\Delta \in \mathcal{D}(\mathsf{sCSP})$ with $[\![P]\!] \stackrel{\hat{a}}{\Longrightarrow} \Delta$ and $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, \Delta)$.*
4. $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\bigoplus_{i \in I} p_i \cdot T_i, P)$ *iff* $o = \sum_{i \in I} p_i o_i$ *for some* $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, P)$.
5. $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\bigsqcap_{i \in I} T_i, P)$ *if for all $i \in I$ there are $q_i \in [0,1]$ and $\Delta_i \in \mathcal{D}(\mathsf{sCSP})$ such that $\sum_{i \in I} q_i = 1$, $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \Delta_i$ and $o = \sum_{i \in I} q_i o_i$ for some $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$.*

Proof: Straightforward, by induction on the structure of P. □

The converse of Lemma 5.41 (5) also holds, as the following lemma says. However, the proof is less straightforward.

Lemma 5.42 *Let P be a* pCSP *process, and T_i be tests. If $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(\bigsqcap_{i \in I} T_i, P)$ then for all $i \in I$ there are $q_i \in [0,1]$ and $\Delta_i \in \mathcal{D}(\mathsf{sCSP})$ with $\sum_{i \in I} q_i = 1$ such that $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \Delta_i$ and $o = \sum_{i \in I} q_i o_i$ for some $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$.*

Proof: Given that the states of our pLTS are sCSP expressions, there exists a well-founded order on the combination of states in sCSP and distributions in $\mathcal{D}(\mathsf{sCSP})$, such that $s \xrightarrow{\alpha} \Delta$ implies that s is larger than Δ, and any distribution is larger than the states in its support. Intuitively, this order corresponds to the usual order on natural numbers if we graphically depict a pLTS as a finite tree (cf. Section 5.2.4) and assign to each node a number to indicate its level in the tree. Let $T = \bigsqcap_{i \in I} T_i$. We prove the following two claims

(a) If s is a state-based process and $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, s)$ then there are some $\{q_i\}_{i \in I}$ with $\sum_{i \in I} q_i = 1$ such that $s \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \Delta_i$, $o = \sum_{i \in I} q_i o_i$, and $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$.

(b) If $\Delta \in \mathcal{D}(\mathsf{sCSP})$ and $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, \Delta)$ then there are some $\{q_i\}_{i \in I}$ with $\sum_{i \in I} q_i = 1$ such that $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \Delta_i$, $o = \sum_{i \in I} q_i o_i$, and $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$.

by simultaneous induction on the order mentioned above, applied to s and Δ.

(a) We have two sub-cases depending on whether s can make an initial τ-move or not.

- If s cannot make a τ-move, that is $s \not\xrightarrow{\tau}$, then the only possible moves from $T \mid_{Act} s$ are τ-moves originating in T; T has no non-τ moves, and any non-τ moves that might be possible for s on its own are inhibited by the alphabet Act of the composition. Suppose $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T, s)$. Then by definition (5.9) there are some $\{q_i\}_{i \in I}$ with $\sum_{i \in I} q_i = 1$ such that $o = \sum_{i \in I} q_i o_i$ and $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, s) = \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \overline{s})$. Obviously we also have $[\![s]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \overline{s}$.

- If s can make one or more τ-moves, then we have $s \xrightarrow{\tau} \Delta'_j$ for $j \in J$, where without loss of generality J can be assumed to be a non-empty finite set disjoint from I, the index set for T. The possible first moves for $T \mid_{Act} s$ are τ-moves either of T or of s, because T cannot make initial non-τ moves and that prevents a proper synchronisation from occurring on the first step. Suppose that $o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, s)$. Then by definition (5.9) there are some $\{p_k\}_{k \in I \cup J}$ with $\sum_{k \in I \cup J} p_k = 1$ and

$$o = \sum_{k \in I \cup J} p_k o'_k \tag{5.10}$$

$$o'_i \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_i, s) \qquad \text{for all } i \in I \tag{5.11}$$

$$o'_j \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, \Delta_j) \qquad \text{for all } j \in J. \tag{5.12}$$

For each $j \in J$, we know by the induction hypothesis that

$$\Delta'_j \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} p_{ji} \cdot \Delta'_{ji} \tag{5.13}$$

$$o'_j = \sum_{i \in I} p_{ji} o'_{ji} \tag{5.14}$$

$$o'_{ji} \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_i, \Delta'_{ji}) \tag{5.15}$$

for some $\{p_{ji}\}_{i \in I}$ with $\sum_{i \in I} p_{ji} = 1$. Let

$$\begin{aligned} q_i &= p_i + \sum_{j \in J} p_j p_{ji} \\ \Delta_i &= \frac{1}{q_i}(p_i \cdot \overline{s} + \sum_{j \in J} p_j p_{ji} \cdot \Delta'_{ji}) \\ o_i &= \frac{1}{q_i}(p_i o'_i + \sum_{j \in J} p_j p_{ji} o'_{ji}) \end{aligned}$$

for each $i \in I$, except that Δ_i and o_i are chosen arbitrarily in case $q_i = 0$. It can be checked by arithmetic that q_i, Δ_i, o_i have the required properties, viz. that $\sum_{i \in I} q_i = 1$, that $o = \sum_{i \in I} q_i o_i$ and that

$$\begin{aligned} s &\stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} p_i \cdot \overline{s} + \sum_{j \in J} p_j \cdot \Delta'_j \\ &\stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} p_i \cdot \overline{s} + \sum_{j \in J} p_j \cdot \sum_{i \in I} p_{ji} \cdot \Delta'_{ji} \qquad \text{by (5.13) and Lemma 5.19} \\ &= \sum_{i \in I} q_i \cdot \Delta_i \ . \end{aligned}$$

Finally, it follows from (5.11) and (5.15) that $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_i, \Delta_i)$ for each $i \in I$.

(b) Let $\lceil \Delta \rceil = \{s_j\}_{j \in J}$ and $r_j = \Delta(s_j)$. W.l.o.g. we may assume that J is a non-empty finite set disjoint from I. Using that $\widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, \Delta) := \mathrm{Exp}_{\Delta} \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, \_)$, if $o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, \Delta)$ then

$$o = \sum_{j \in J} r_j o'_j \tag{5.16}$$

$$o'_j \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T, s_j) \tag{5.17}$$

For each $j \in J$, we know by the induction hypothesis that

$$s_j \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_{ji} \cdot \Delta'_{ji} \tag{5.18}$$

$$o'_j = \sum_{i \in I} q_{ji} o'_{ji} \tag{5.19}$$

$$o'_{ji} \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_i, \Delta'_{ji}) \tag{5.20}$$

for some $\{q_{ji}\}_{i \in I}$ with $\sum_{i \in I} q_{ji} = 1$. Thus let

$$q_i = \sum_{j \in J} r_j q_{ji}$$

$$\Delta_i = \frac{1}{q_i} \sum_{j \in J} r_j q_{ji} \cdot \Delta'_{ji}$$

$$o_i = \frac{1}{q_i} \sum_{j \in J} r_j q_{ji} o'_{ji}$$

again choosing Δ_i and o_i arbitrarily in case $q_i = 0$. As in the first case, it can be shown by arithmetic that the collection r_i, Δ_i, o_i has the required properties.

□

5.8 Resolution-based testing

A *probabilistic automaton* consists of a pLTS $\langle S, L, \rightarrow \rangle$ and a distribution Δ° over S. Since we only consider probabilistic automata with $L = Act_\tau \cup \Omega$, we omit it and write a probabilistic automaton simply as a triple $\langle S, \Delta^\circ, \rightarrow \rangle$ and call Δ° the *initial distribution* of the automaton. The operational semantics of a pCSP^Ω process P can thus be viewed as a probabilistic automaton with initial distribution $\Delta^\circ := [\![P]\!]$. States in a probabilistic automata that are not reachable from the initial distribution are generally considered irrelevant and can be omitted.

A probabilistic automaton is called *finite* if there exists a function $depth : S \cup \mathcal{D}(S) \rightarrow \mathbb{N}$ such that $s \in \lceil \Delta \rceil$ implies $depth(s) < depth(\Delta)$ and $s \xrightarrow{\alpha} \Delta$ implies $depth(s) > depth(\Delta)$. Finite probabilistic automata can be drawn as explained at the end of Section 5.2.4.

A *fully probabilistic automaton* is one in which each state enables at most one action, and (general) probabilistic automata can be "resolved" into fully probabilistic automata by pruning away multiple action-choices until only single choices are left, possibly introducing some linear combinations in the process. We define this formally for probabilistic automata representing pCSP^Ω expressions.

Definition 5.43 *[DvGMZ07] A* resolution *of a distribution $\Delta^\circ \in \mathcal{D}(\mathsf{sCSP}^\Omega)$ is a fully probabilistic automaton $\langle R, \Theta^\circ, \rightarrow \rangle$ such that there is a resolving function $f : R \rightarrow \mathsf{sCSP}^\Omega$ which satisfies:*

(i) $f(\Theta^\circ) = \Delta^\circ$

(ii) if $r \xrightarrow{\alpha} \Theta$ *then* $f(r) \xrightarrow{\alpha} f(\Theta)$

(iii) if $r \not\rightarrow$ *then* $f(r) \not\rightarrow$

where $f(\Theta)$ is the distribution defined by $f(\Theta)(s) := \sum_{f(r)=s} \Theta(r)$.

Note that resolutions of distributions $\Delta^\circ \in \mathcal{D}(\mathsf{sCSP}^\Omega)$ are always finite. We define a function which yields the probability that a given fully probabilistic automaton will start with a particular sequence of actions.

Definition 5.44 *[DvGMZ07] Given a fully probabilistic automaton* $\mathrm{R} = \langle R, \Delta^\circ, \rightarrow \rangle$*, the probability that* R *follows the sequence of actions* $\sigma \in \Sigma^*$ *from its initial distribution is given by* $\Pr_{\mathrm{R}}(\sigma, \Delta^\circ)$*, where* $\Pr_{\mathrm{R}} : \Sigma^* \times R \to [0,1]$ *is defined inductively by*

$$\Pr_{\mathrm{R}}(\varepsilon, r) := 1 \quad \textit{and} \quad \Pr_{\mathrm{R}}(\alpha\sigma, r) := \begin{cases} \Pr_{\mathrm{R}}(\sigma, \Delta) & \textit{if } r \xrightarrow{\alpha} \Delta \\ 0 & \textit{otherwise} \end{cases}$$

and $\Pr_{\mathrm{R}}(\sigma, \Delta) := \mathit{Exp}_\Delta(\Pr_{\mathrm{R}}(\sigma, \_)) = \sum_{r \in \lceil \Delta \rceil} \Delta(r) \cdot \Pr_{\mathrm{R}}(\sigma, r)$*. Here* ε *denotes the empty sequence of actions and* $\alpha\sigma$ *the sequence starting with* $\alpha \in \Sigma$ *and continuing with* $\sigma \in \Sigma^*$*. The value* $\Pr_{\mathrm{R}}(\sigma, r)$ *is the probability that* R *proceeds with sequence* σ *from state* r*.*

Now let $\Sigma^{*\alpha}$ *be the set of finite sequences in* Σ^* *that contain* α *exactly once, and that at the end. Then the probability that the fully probabilistic automaton* R *ever performs an action* α *is given by* $\sum_{\sigma \in \Sigma^{*\alpha}} \Pr_{\mathrm{R}}(\sigma, \Delta^\circ)$*.*

We recall the results-gathering function $\mathbb{W}$ given in Definition 5 of [DvGMZ07].

Definition 5.45 *For a fully probabilistic automaton* R*, let its* success tuple $\mathbb{W}(\mathrm{R}) \in [0,1]^\Omega$ *be such that* $\mathbb{W}(\mathrm{R})(\omega)$ *is the probability that* R *ever performs the action* ω*.*

Then for a distribution $\Delta^\circ \in \mathcal{D}(\mathsf{sCSP}^\Omega)$ *we define the* set *of its success tuples to be those resulting as above from all its resolutions separately:*

$$\mathbb{W}(\Delta^\circ) \;:=\; \{\mathbb{W}(\mathrm{R}) \mid \mathrm{R} \textit{ is a resolution of } \Delta^\circ\}.$$

We relate these sets of tuples to Definition 5.35, in which similar sets are produced "all at once," that is without introducing resolutions first. In fact we will find that they are the same. Note that Definition 5.35 of $\widehat{\mathbb{V}}^\Omega_\downarrow$ extends smoothly to states and distributions in probabilistic automata. When applied to fully probabilistic automata, $\widehat{\mathbb{V}}^\Omega_\downarrow$ always yields singleton sets, which we will loosely identify with their unique members; thus when we write $\widehat{\mathbb{V}}^\Omega_\downarrow(\Delta)(\omega)$ with Δ a distribution in a fully probabilistic automaton, we actually mean the ω-component of the unique element of $\widehat{\mathbb{V}}^\Omega_\downarrow(\Delta)$.

Lemma 5.46 *If* $\mathrm{R} = \langle R, \Delta^\circ, \rightarrow \rangle$ *is a finite fully probabilistic automaton, then*

(1) $\widehat{\mathbb{V}}^\Omega_\downarrow(\Delta) = \widehat{\mathbb{V}}^\Omega_\downarrow(\Delta)$ *for all* $\Delta \in \mathcal{D}(R)$*, and*

(2) $\mathbb{W}(\mathrm{R}) = \widehat{\mathbb{V}}^\Omega_\downarrow(\Delta^\circ)$*.*

Proof: (1) is immediate: since the automaton is fully probabilistic, convex closure has no effect. For (2) we need to show that for all $\omega \in \Omega$ we have $\mathbb{W}(\mathrm{R})(\omega) = \widehat{\mathbb{V}}^\Omega_\downarrow(\Delta^\circ)(\omega)$, i.e. that $\sum_{\sigma \in \Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, \Delta^\circ) = (\widehat{\mathbb{V}}^\Omega_\downarrow(\Delta^\circ))(\omega)$. So let $\omega \in \Omega$. We show

$$\sum_{\sigma \in \Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, \Delta) = \widehat{\mathbb{V}}^\Omega_\downarrow(\Delta)(\omega) \qquad \text{and} \qquad \sum_{\sigma \in \Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, r) = \widehat{\mathbb{V}}^\Omega_\downarrow(r)(\omega) \tag{5.21}$$

for all $\Delta \in \mathcal{D}(R)$ and $r \in R$, by simultaneous induction on the depths of Δ and r.

- In the base case r has no enabled actions. Then $\forall i : \sum_{\sigma \in \Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, r) = 0$ and $\widehat{\mathbb{V}}^\Omega_\downarrow(r) = \vec{0}$, so $\widehat{\mathbb{V}}^\Omega_\downarrow(r)(\omega) = 0$.
- Now suppose there is a transition $r \xrightarrow{\alpha} \Delta$ for some action α and distribution Δ. There are two possibilities:
  - $\alpha = \omega$. We then have $\widehat{\mathbb{V}}^\Omega_\downarrow(s)(\omega) = 1$. Now for any finite non-empty sequence σ without any occurrence of ω we have $\Pr_{\mathrm{R}}(\sigma\omega, r) = 0$. Thus $\sum_{\sigma \in \Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, r) = \Pr_{\mathrm{R}}(\omega, r) = 1$ as required.

– $\alpha \neq \omega$. Since $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) = \alpha!\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta)$, we have $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r)(\omega) = \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta)(\omega)$. On the other hand, $\Pr_{\mathrm{R}}(\beta\sigma, r) = 0$ for $\beta \neq \alpha$. Therefore

$$\begin{array}{rcll} \sum_{\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, r) & = & \sum_{\alpha\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\alpha\sigma, r) & \\ & = & \sum_{\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\alpha\sigma, r) & \\ & = & \sum_{\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, \Delta) & \\ & = & \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta)(\omega) & \text{by induction} \\ & = & \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r)(\omega)\ . & \end{array}$$

- Finally, $\sum_{\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, \Delta) = \sum_{\sigma\in\Sigma^{*\omega}} \mathrm{Exp}_{\Delta}(\Pr_{\mathrm{R}}(\sigma, \_)) = \mathrm{Exp}_{\Delta}(\sum_{\sigma\in\Sigma^{*\omega}} \Pr_{\mathrm{R}}(\sigma, \_))$ $= \mathrm{Exp}_{\Delta}(\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\_)(\omega)) = \mathrm{Exp}_{\Delta}(\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\_))(\omega) = \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta)(\omega)$.

□

Now we look more closely at the interaction of $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}$ and resolutions.

Lemma 5.47 *Let $\Delta^{\circ} \in \mathcal{D}(\mathsf{sCSP}^{\Omega})$.*

(1) If $\langle R, \Theta^{\circ}, \rightarrow\rangle$ is a resolution of Δ°, then $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta^{\circ}) \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta^{\circ})$.

(2) If $o \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta^{\circ})$ then there is a resolution $\langle R, \Theta^{\circ}, \rightarrow\rangle$ of Δ° such that $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta^{\circ}) = o$.

Proof:

(1) Let $\langle R, \Theta^{\circ}, \rightarrow\rangle$ be a resolution of Δ° with resolving function f. We observe that for any $\Theta \in \mathcal{D}(R)$ we have

$$\forall r \in \lceil\Theta\rceil : \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r)) \text{ implies } \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta) \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(\Theta)) \tag{5.22}$$

because

$$\begin{array}{rcl} \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta) & = & \sum_{r\in\lceil\Theta\rceil} \Theta(r) \cdot \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) \\ & \in & \sum_{r\in\lceil\Theta\rceil} \Theta(r) \cdot \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r)) \\ & = & \sum_{s\in\lceil f(\Theta)\rceil} f(\Theta)(s) \cdot \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(s) \\ & = & \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(\Theta))\ . \end{array}$$

We now prove by induction on *depth*(r) that $\forall r \in T : \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r))$, from which the required result follows in view of (5.22) and the fact that $f(\Theta^{\circ}) = \Delta^{\circ}$.

- In the base case we have $r \not\rightarrow$, which implies $f(r) \not\rightarrow$. Therefore, we have $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) = \vec{0} \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r))$.
- Otherwise r has a transition $r \xrightarrow{\alpha} \Theta$ for some α and Θ. By induction we have $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r') \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r'))$ for all $r' \in \lceil\Theta\rceil$. Using (5.22) we get $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta) \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(\Theta))$. Now

$$\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(r) \ = \ \alpha!\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta) \ \in \ \alpha!\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(\Theta)) \ \subseteq \ \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(f(r))$$

where the last step follows from the fact that $f(r) \xrightarrow{\alpha} f(\Theta)$ is one of the transitions of $f(r)$.

(2) This clause is proved by induction on *depth*(Δ°). First consider the special case that Δ° is a point distribution on some state s.

- In the base case we have $s \not\rightarrow$. The probabilistic automaton $\langle\{s\}, \overline{s}, \emptyset\rangle$ is a resolution of $\Delta^{\circ} = \overline{s}$ with the resolving function being the identity. Clearly, this resolution satisfies our requirement.
- Otherwise there is a finite, non-empty index set I such that $s \xrightarrow{\alpha_i} \Delta_i$ for some actions α_i and distributions Δ_i. If $o \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta^{\circ}) = \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(s)$, then by the definition of $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}$ we have $o = \sum_{i\in I} p_i \cdot \alpha_i!o_i$ with $o_i \in \widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Delta_i)$ and $\sum_{i\in I} p_i = 1$ for some $p_i \in [0, 1]$. By induction, for each $i \in I$ there is a resolution $\langle R_i, \Theta_i^{\circ}, \rightarrow_i\rangle$ of Δ_i with resolving function f_i such that $\widehat{\mathbb{V}}^{\Omega}_{\updownarrow}(\Theta_i^{\circ}) = o_i$. Without loss of generality, we assume that R_i is disjoint from R_j for $i \neq j$, as well as from $\{r_i \mid i \in I\}$. We now construct a fully probabilistic automaton $\langle R, \Theta^{\circ}, \rightarrow'\rangle$ as follows:

- $R := \{r_i \mid i \in I\} \cup \bigcup_{i\in I} R_i$
- $\Theta^\circ := \sum_{i\in I} p_i \cdot \overline{r_i}$
- $\rightarrow' := \{r_i \xrightarrow{\alpha_i} \Theta_i^\circ \mid i \in I\} \cup \bigcup_{i\in I} \rightarrow_i$.

This automaton is a resolution of $\Delta^\circ = \overline{s}$ with resolving function f defined by

$$f(r) = \begin{cases} s & \text{if } r = r_i \text{ for } i \in I \\ f_i(r) & \text{if } r \in R_i \text{ for } i \in I. \end{cases}$$

The resolution thus constructed satisfies our requirement because

$$\begin{array}{rcl} \widehat{\mathbb{V}}_\updownarrow^\Omega(\Theta^\circ) & = & \widehat{\mathbb{V}}_\updownarrow^\Omega(\sum_{i\in I} p_i \cdot \overline{r_i}) \\ & = & \sum_{i\in I} p_i \cdot \widehat{\mathbb{V}}_\updownarrow^\Omega(r_i) \\ & = & \sum_{i\in I} p_i \cdot \alpha_i ! \widehat{\mathbb{V}}_\updownarrow^\Omega(\Theta_i^\circ) \\ & = & \sum_{i\in I} p_i \cdot \alpha_i ! o_i \\ & = & o\ . \end{array}$$

We now consider the general case that Δ° is a proper distribution with $\lceil \Delta^\circ \rceil = \{s_j \mid j \in J\}$ for some finite index set J. Using the reasoning in the above special case, we have a resolution $\langle R_j, \Theta_j^\circ, \rightarrow_j \rangle$ of each distribution $\overline{s_j}$. Without loss of generality, we assume that R_j is disjoint from R_k for $j \neq k$. Consider the probabilistic automaton $\langle \bigcup_{j\in J} R_j, \sum_{j\in J} \Delta^\circ(s_j) \cdot \Theta_j^\circ, \bigcup_{j\in J} \rightarrow_j \rangle$. It is a resolution of Δ° satisfying our requirement. If $o \in \widehat{\mathbb{V}}_\updownarrow^\Omega(\Delta^\circ)$ then $o = \sum_{j\in J} \Delta^\circ(s_j) \cdot o_j$ with $o_j \in \widehat{\mathbb{V}}_\updownarrow^\Omega(s_j)$. Since $o_j = \widehat{\mathbb{V}}_\updownarrow^\Omega(\Theta_j^\circ)$, we have $o = \widehat{\mathbb{V}}_\updownarrow^\Omega(\sum_{j\in J} \Delta^\circ(s_j) \cdot \Theta_j^\circ)$.

□

We can now give the result relied on in Section 5.7.

Proposition 5.48 *Let $\Delta^\circ \in \mathcal{D}(\mathsf{sCSP}^\Omega)$. Then we have that $\mathbb{W}(\Delta^\circ) = \widehat{\mathbb{V}}_\updownarrow^\Omega(\Delta^\circ)$.*

Proof: Combine Lemmas 5.46 and 5.47. □

5.9 Modal logic

In this section we present logical characterisations $\sqsubseteq^{\mathcal{L}}$ and $\sqsubseteq^{\mathcal{F}}$ of our testing preorders. Besides their intrinsic interest, these logical preorders also serves as a stepping stone in proving Theorem 5.33. In this section we show that the logical preorders are sound w.r.t. the simulation and failure simulation preorders, and hence w.r.t. the testing preorders; in the next section we establish completeness. To start, we define a set $\mathcal{F}$ of modal formulae, inductively, as follows:

- $\mathbf{ref}(X) \in \mathcal{F}$ when $X \subseteq Act$,
- $\langle a \rangle \phi \in \mathcal{F}$ when $\phi \in \mathcal{F}$ and $a \in Act$,
- $\bigwedge_{i\in I} \phi_i \in \mathcal{F}$ when $\phi_i \in \mathcal{F}$ for all $i \in I$, with I finite,
- and $\bigoplus_{i\in I} p_i \cdot \phi_i \in \mathcal{F}$ when $p_i \in [0,1]$ and $\phi_i \in \mathcal{F}$ for all $i \in I$, with I a finite index set, and $\sum_{i\in I} p_i = 1$.

We often write $\phi_1 \wedge \phi_2$ for $\bigwedge_{i\in\{1,2\}} \phi_i$ and $\top$ for $\bigwedge_{i\in\emptyset} \phi_i$.

The *satisfaction relation* $\models \subseteq \mathcal{D}(\mathsf{sCSP}) \times \mathcal{F}$ is given by:

- $\Delta \models \mathbf{ref}(X)$ iff there is a Δ' with $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Delta'$ and $\Delta' \stackrel{X}{\not\longrightarrow}$,
- $\Delta \models \langle a \rangle \phi$ iff there is a Δ' with $\Delta \stackrel{\hat{a}}{\Longrightarrow} \Delta'$ and $\Delta' \models \phi$,
- $\Delta \models \bigwedge_{i\in I} \phi_i$ iff $\Delta \models \phi_i$ for all $i \in I$

- and $\Delta \models \bigoplus_{i\in I} p_i \cdot \phi_i$ iff there are $\Delta_i \in \mathcal{D}(\mathsf{sCSP})$, for all $i \in I$, with $\Delta_i \models \phi_i$, such that $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i\in I} p_i \cdot \Delta_i$.

Let $\mathcal{L}$ be the subclass of $\mathcal{F}$ obtained by skipping the $\mathbf{ref}(X)$ clause. We write $P \sqsubseteq^{\mathcal{L}} Q$ just when $[\![P]\!] \models \phi$ implies $[\![Q]\!] \models \phi$ for all $\phi \in \mathcal{L}$, and $P \sqsubseteq^{\mathcal{F}} Q$ just when $[\![P]\!] \models \phi$ is implied by $[\![Q]\!] \models \phi$ for all $\phi \in \mathcal{F}$. (Note the opposing directions.)

In order to obtain the main result of this section, Theorem 5.52, we introduce the following tool.

Definition 5.49 *The $\mathcal{F}$-*characteristic formula ϕ_s *or* ϕ_Δ *of a process* $s \in \mathsf{sCSP}$ *or* $\Delta \in \mathcal{D}(\mathsf{sCSP})$ *is defined inductively:*

- $\phi_s := \bigwedge_{s \xrightarrow{a}_\Delta} \langle a \rangle \phi_\Delta \wedge \mathbf{ref}(\{a \mid s \stackrel{a}{\not\longrightarrow}\})$ *if* $s \stackrel{\tau}{\not\longrightarrow}$,
- $\phi_s := \bigwedge_{s \xrightarrow{a}_\Delta} \langle a \rangle \phi_\Delta \wedge \bigwedge_{s \xrightarrow{\tau}_\Delta} \phi_\Delta$ *otherwise,*
- $\phi_\Delta := \bigoplus_{s \in \lceil \Delta \rceil} \Delta(s) \cdot \phi_s$.

*Here the conjunctions $\bigwedge_{s \xrightarrow{a}_\Delta}$ range over suitable pairs a, Δ, and $\bigwedge_{s \xrightarrow{\tau}_\Delta}$ ranges over suitable Δ. The $\mathcal{L}$-*characteristic formulae ψ_s *and* ψ_Δ *are defined likewise, but omitting the conjuncts* $\mathbf{ref}(\{a \mid s \stackrel{a}{\not\longrightarrow}\})$.

Write $\phi \Rrightarrow \psi$ with $\phi, \psi \in \mathcal{F}$ if for each distribution Δ one has $\Delta \models \phi$ implies $\Delta \models \psi$. Then it is easy to see that $\phi_{\overline{s}} \Longleftrightarrow \phi_s$ and $\bigwedge_{i\in I} \phi_i \Rrightarrow \phi_i$ for any $i \in I$; furthermore, the following property can be established by an easy inductive proof.

Lemma 5.50 *For any $\Delta \in \mathcal{D}(\mathsf{sCSP})$ we have $\Delta \models \phi_\Delta$, as well as $\Delta \models \psi_\Delta$.* □

It and the following lemma help to establish Theorem 5.52.

Lemma 5.51 *For any processes $P, Q \in \mathsf{pCSP}$ we have that $[\![P]\!] \models \phi_{[\![Q]\!]}$ implies $P \sqsubseteq_{FS} Q$, and likewise that $[\![Q]\!] \models \psi_{[\![P]\!]}$ implies $P \sqsubseteq_S Q$.*

Proof: To establish the first statement, we define the relation $\mathcal{R}$ by $s \mathcal{R} \Theta$ iff $\Theta \models \phi_s$; to show that it is a failure simulation we first prove the following technical result:

$$\Theta \models \phi_\Delta \;\text{ implies }\; \exists \Theta' : \Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta' \wedge \Delta \;\overline{\mathcal{R}}\; \Theta'. \tag{5.23}$$

Suppose $\Theta \models \phi_\Delta$ with $\phi_\Delta = \bigoplus_{i\in I} p_i \cdot \phi_{s_i}$, so that we have $\Delta = \sum_{i\in I} p_i \cdot \overline{s_i}$ and for all $i \in I$ there are $\Theta_i \in \mathcal{D}(\mathsf{sCSP})$ with $\Theta_i \models \phi_{s_i}$ such that $\Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta'$ with $\Theta' := \sum_{i\in I} p_i \cdot \Theta_i$. Since $s_i \mathcal{R} \Theta_i$ for all $i \in I$ we have $\Delta \;\overline{\mathcal{R}}\; \Theta'$.

Now we show that $\mathcal{R}$ is a failure simulation.

- Suppose $s \mathcal{R} \Theta$ and $s \xrightarrow{\tau} \Delta$. Then from Definition 5.49 we have $\phi_s \Rrightarrow \phi_\Delta$, so that $\Theta \models \phi_\Delta$. Applying (5.23) gives us $\Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta'$ with $\Delta \;\overline{\mathcal{R}}\; \Theta'$ for some Θ'.
- Suppose $s \mathcal{R} \Theta$ and $s \xrightarrow{a} \Delta$ with $a \in Act$. Then $\phi_s \Rrightarrow \langle a \rangle \phi_\Delta$, so $\Theta \models \langle a \rangle \phi_\Delta$. Hence $\exists \Theta'$ with $\Theta \stackrel{\hat{a}}{\Longrightarrow} \Theta'$ and $\Theta' \models \phi_\Delta$. Again apply (5.23).
- Suppose $s \mathcal{R} \Theta$ and $s \stackrel{X}{\not\longrightarrow}$ with $X \subseteq A$. Then $\phi_s \Rrightarrow \mathbf{ref}(X)$, so $\Theta \models \mathbf{ref}(X)$. Hence $\exists \Theta'$ with $\Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta'$ and $\Theta' \stackrel{X}{\not\longrightarrow}$.

Thus $\mathcal{R}$ is indeed a failure simulation. By our assumption $[\![P]\!] \models \phi_{[\![Q]\!]}$, using (5.23), there exists a Θ' such that $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Theta'$ and $[\![Q]\!] \;\overline{\mathcal{R}}\; \Theta'$, which gives $P \sqsubseteq_{FS} Q$ via Definition 5.15.

To establish the second statement, define the relation $\mathcal{S}$ by $s \mathcal{S} \Theta$ iff $\Theta \models \psi_s$; exactly as above one obtains

$$\Theta \models \psi_\Delta \;\text{ implies }\; \exists \Theta' : \Theta \stackrel{\hat{\tau}}{\Longrightarrow} \Theta' \wedge \Delta \;\overline{\mathcal{S}}\; \Theta'. \tag{5.24}$$

Just as above it follows that $\mathcal{S}$ is a simulation. By the assumption $[\![Q]\!] \models \phi_{[\![P]\!]}$, using (5.24), there exists a Θ' such that $[\![Q]\!] \stackrel{\hat{\tau}}{\Longrightarrow} \Theta'$ and $[\![P]\!] \;\overline{\mathcal{S}}\; \Theta'$. Hence $P \sqsubseteq_S Q$ via Definition 5.15. □

Theorem 5.52

1. *If* $P \sqsubseteq^{\mathcal{L}} Q$ *then* $P \sqsubseteq_S Q$.
2. *If* $P \sqsubseteq^{\mathcal{F}} Q$ *then* $P \sqsubseteq_{FS} Q$.

Proof: Suppose $P \sqsubseteq^{\mathcal{F}} Q$. By Lemma 5.50 we have $[\![Q]\!] \models \phi_{[\![Q]\!]}$ and hence $[\![P]\!] \models \phi_{[\![Q]\!]}$. Lemma 5.51 gives $P \sqsubseteq_{FS} Q$.

For (1), assuming $P \sqsubseteq^{\mathcal{L}} Q$, we have $[\![P]\!] \models \psi_{[\![P]\!]}$, hence $[\![Q]\!] \models \psi_{[\![P]\!]}$, and thus $P \sqsubseteq_S Q$. □

5.10 Characteristic tests

Our final step towards Theorem 5.33 is taken in this section, where we show that every modal formula ϕ can be characterised by a vector-based test T_ϕ with the property that any pCSP process satisfies ϕ just when it passes the test T_ϕ.

Lemma 5.53 *For every* $\phi \in \mathcal{F}$ *there exists a pair* (T_ϕ, v_ϕ) *with* T_ϕ *an* Ω*-test and* $v_\phi \in [0,1]^\Omega$*, such that*

$$\Delta \models \phi \quad \textit{iff} \quad \exists o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_\phi, \Delta) : \ o \leq v_\phi \tag{5.25}$$

for all $\Delta \in \mathcal{D}(\mathsf{sCSP})$*, and in case* $\phi \in \mathcal{L}$ *we also have*

$$\Delta \models \phi \quad \textit{iff} \quad \exists o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_\phi, \Delta) : \ o \geq v_\phi \,. \tag{5.26}$$

T_ϕ is called a *characteristic test* of ϕ and v_ϕ its *target value*.

Proof: First of all note that if a pair (T_ϕ, v_ϕ) satisfies the requirements above, then any pair obtained from (T_ϕ, v_ϕ) by bijectively renaming the elements of Ω also satisfies these requirements. Hence a characteristic test can always be chosen in such a way that there is a success action $\omega \in \Omega$ that does not occur in (the finite) T_ϕ. Moreover, any countable collection of characteristic tests can be assumed to be , meaning that no $\omega \in \Omega$ occurs in two different elements of the collection.

The required characteristic tests and target values are obtained as follows.

- Let $\phi = \top$. Take $T_\phi := \omega$ for some $\omega \in \Omega$, and $v_\phi := \vec{\omega}$.
- Let $\phi = \mathbf{ref}(X)$ with $X \subseteq Act$. Take $T_\phi := \Box_{a \in X}\, a.\omega$ for some $\omega \in \Omega$, and $v_\phi := \vec{0}$.
- Let $\phi = \langle a \rangle \psi$. By induction, ψ has a characteristic test T_ψ with target value v_ψ. Take $T_\phi := \omega \,\Box\, a.T_\psi$ where $\omega \in \Omega$ does not occur in T_ψ, and $v_\phi := v_\psi$.
- Let $\phi = \bigwedge_{i \in I} \phi_i$ with I a finite and non-empty index set. Choose a Ω-disjoint family $(T_i, v_i)_{i \in I}$ of characteristic tests T_i with target values v_i for each ϕ_i. Furthermore, let $p_i \in (0,1]$ for $i \in I$ be chosen arbitrarily such that $\sum_{i \in I} p_i = 1$. Take $T_\phi := \bigoplus_{i \in I} p_i \cdot T_i$ and $v_\phi := \sum_{i \in I} p_i v_i$.
- Let $\phi = \bigoplus_{i \in I} p_i \cdot \phi_i$. Choose a Ω-disjoint family $(T_i, v_i)_{i \in I}$ of characteristic tests T_i with target values v_i for each ϕ_i, such that there are distinct success actions ω_i for $i \in I$ that do not occur in any of those tests. Let $T_i' := T_i \,{}_{\frac{1}{2}}\!\oplus\, \omega_i$ and $v_i' := \frac{1}{2} v_i + \frac{1}{2}\vec{\omega_i}$. Note that for all $i \in I$ also T_i' is a characteristic test of ϕ_i with target value v_i'. Take $T_\phi := \bigsqcap_{i \in I} T_i'$ and $v_\phi := \sum_{i \in I} p_i v_i'$.

Note that $v_\phi(\omega) = 0$ whenever $\omega \in \Omega$ does not occur in T_ϕ. By induction on ϕ we now check (5.25) above.

- Let $\phi = \top$. For all $\Delta \in \mathcal{D}(\mathsf{sCSP})$ we have $\Delta \models \phi$ as well as $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_\phi, \Delta) : \ o \leq v_\phi$, using Lemma 5.41(1).
- Let $\phi = \mathbf{ref}(X)$ with $X \subseteq Act$. Suppose $\Delta \models \phi$. Then there is a Δ' with $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Delta'$ and $\Delta' \stackrel{X}{\not\longrightarrow}$. By Lemma 5.41(2), $\vec{0} \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_\phi, \Delta)$.

  Now suppose $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\updownarrow}(T_\phi, \Delta) : \ o \leq v_\phi$. This implies $o = \vec{0}$, so by Lemma 5.41(2) there is a Δ' with $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \Delta'$ and $\Delta' \stackrel{X}{\not\longrightarrow}$. Hence $\Delta \models \phi$.

- Let $\phi = \langle a \rangle \psi$ with $a \in Act$. Suppose $\Delta \models \phi$. Then there is a Δ' with $\Delta \stackrel{\hat{a}}{\Longrightarrow} \Delta'$ and $\Delta' \models \psi$. By induction, $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\psi, \Delta') :\ o \leq v_\psi$. By Lemma 5.41(3), $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta)$.

  Now suppose $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta) :\ o \leq v_\phi$. This implies $o(\omega) = 0$, so by Lemma 5.41(3) there is a Δ' with $\Delta \stackrel{\hat{a}}{\Longrightarrow} \Delta'$ and $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\psi, \Delta')$. By induction, $\Delta' \models \psi$, so $\Delta \models \phi$.

- Let $\phi = \bigwedge_{i \in I} \phi_i$ with I a finite and non-empty index set. Suppose $\Delta \models \phi$. Then $\Delta \models \phi_i$ for all $i \in I$, and hence, by induction, $\exists o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta) :\ o_i \leq v_i$. Thus $o := \sum_{i \in I} p_i o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta)$ by Lemma 5.41(4), and $o \leq v_\phi$.

  Now suppose $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta) :\ o \leq v_\phi$. Then, using Lemma 5.41(4), $o = \sum_{i \in I} p_i o_i$ for certain $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta)$. Note that $(T_i)_{i \in I}$ is an Ω-disjoint family of tests. One has $o_i \leq v_i$ for all $i \in I$, for if $o_i(\omega) > v_i(\omega)$ for some $i \in I$ and $\omega \in \Omega$, then ω must occur in T_i and hence cannot occur in T_j for $j \neq i$. This implies $v_j(\omega) = 0$ for all $j \neq i$ and thus $o(\omega) > v_\phi(\omega)$, in contradiction with the assumption. By induction, $\Delta \models \phi_i$ for all $i \in I$, and hence $\Delta \models \phi$.

- Let $\phi = \bigoplus_{i \in I} p_i \cdot \phi_i$. Suppose $\Delta \models \phi$. Then for all $i \in I$ there are $\Delta_i \in \mathcal{D}(\mathsf{sCSP})$ with $\Delta_i \models \phi_i$ such that $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} p_i \cdot \Delta_i$. By induction, there are $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$ with $o_i \leq v_i$. Hence, there are $o'_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T'_i, \Delta_i)$ with $o'_i \leq v'_i$. Thus $o := \sum_{i \in I} p_i o'_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta)$ by Lemma 5.41(5), and $o \leq v_\phi$.

  Now suppose $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta) :\ o \leq v_\phi$. Then, by Lemma 5.42, there are $q \in \mathcal{D}(I)$ and Δ_i, for $i \in I$, such that $\Delta \stackrel{\hat{\tau}}{\Longrightarrow} \sum_{i \in I} q_i \cdot \Delta_i$ and $o = \sum_{i \in I} q_i o'_i$ for some $o'_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T'_i, \Delta_i)$. Now $\forall i :\ o'_i(\omega_i) = v'_i(\omega_i) = \frac{1}{2}$, so, using that $(T_i)_{i \in I}$ is an Ω-disjoint family of tests, $\frac{1}{2} q_i = q_i o'_i(\omega_i) = o(\omega_i) \leq v_\phi(\omega_i) = p_i v'_i(\omega_i) = \frac{1}{2} p_i$. As $\sum_{i \in I} q_i = \sum_{i \in I} p_i = 1$, it must be that $q_i = p_i$ for all $i \in I$. Exactly as in the previous case one obtains $o'_i \leq v'_i$ for all $i \in I$. Given that $T'_i = T_i \, {}_{\frac{1}{2}}\!\oplus \omega_i$, using Lemma 5.41(4), it must be that $o' = \frac{1}{2} o_i + \frac{1}{2} \vec{\omega_i}$ for some $o_i \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_i, \Delta_i)$ with $o_i \leq v_i$. By induction, $\Delta_i \models \phi_i$ for all $i \in I$, and hence $\Delta \models \phi$.

In case $\phi \in \mathcal{L}$, the formula cannot be of the form $\mathbf{ref}(X)$. Then a straightforward induction yields that $\sum_{\omega \in \Omega} v_\phi(\omega) = 1$ and for all $\Delta \in \mathcal{D}(\mathsf{pCSP})$ and $o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, \Delta)$ we have $\sum_{\omega \in \Omega} o(\omega) = 1$. Therefore, $o \leq v_\phi$ iff $o \geq v_\phi$ iff $o = v_\phi$, yielding (5.26). □

Theorem 5.54

1. *If* $P \, \widehat{\sqsubseteq}^{\Omega}_{pmay} \, Q$ *then* $P \sqsubseteq^{\mathcal{L}} Q$.
2. *If* $P \, \widehat{\sqsubseteq}^{\Omega}_{pmust} \, Q$ *then* $P \sqsubseteq^{\mathcal{F}} Q$.

Proof: Suppose $P \, \widehat{\sqsubseteq}^{\Omega}_{\text{pmust}} \, Q$ and $[\![Q]\!] \models \phi$ for some $\phi \in \mathcal{F}$. Let T_ϕ be a characteristic test of ϕ with target value v_ϕ. Then Lemma 5.53 yields $\exists o \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, [\![Q]\!]) : o \leq v_\phi$, and hence, given that $P \, \widehat{\sqsubseteq}^{\Omega}_{\text{pmust}} \, Q$ and $\widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, [\![R]\!]) = \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, R)$ for any $R \in \mathsf{pCSP}$, by the Smyth preorder we have $\exists o' \in \widehat{\mathcal{A}}^{\Omega}_{\downarrow}(T_\phi, [\![P]\!]) :\ o' \leq v_\phi$. Thus $[\![P]\!] \models \phi$.

The may-case goes likewise, via the Hoare preorder. □

Combining Theorems 5.40, 5.54 and 5.52, we obtain Theorem 5.33, the goal we set ourselves in Section 5.6. Thus, with Theorems 5.27 and 5.31 and Proposition 5.32, we have shown that the may preorder coincides with simulation and that the must preorder coincides with failure simulation. These results also imply the converse of both statements in Theorem 5.54, and thus that the logics $\mathcal{L}$ and $\mathcal{F}$ give logical characterisations of the simulation and failure simulation preorders $\sqsubseteq_S$ and $\sqsubseteq_{FS}$. □

5.11 Equational theories

Having settled the problem of characterising the may preorder in terms of simulation, and the must preorder in terms of failure simulation, we now turn to complete axiomatisations of the preorders.

$$
\begin{array}{lrcl}
(\mathbf{P1}) & P \,{}_p\!\oplus P & = & P \\
(\mathbf{P2}) & P \,{}_p\!\oplus Q & = & Q \,{}_{1-p}\!\oplus P \\
(\mathbf{P3}) & (P \,{}_p\!\oplus Q) \,{}_q\!\oplus R & = & P \,{}_{p\cdot q}\!\oplus (Q \,{}_{\frac{(1-p)\cdot q}{1-p\cdot q}}\!\oplus R) \\
(\mathbf{I1}) & P \sqcap P & = & P \\
(\mathbf{I2}) & P \sqcap Q & = & Q \sqcap P \\
(\mathbf{I3}) & (P \sqcap Q) \sqcap R & = & P \sqcap (Q \sqcap R) \\
(\mathbf{E1}) & P \,\Box\, \mathbf{0} & = & P \\
(\mathbf{E2}) & P \,\Box\, Q & = & Q \,\Box\, P \\
(\mathbf{E3}) & (P \,\Box\, Q) \,\Box\, R & = & P \,\Box\, (Q \,\Box\, R) \\
(\mathbf{EI}) & a.P \,\Box\, a.Q & = & a.P \sqcap a.Q \\
(\mathbf{D1}) & P \,\Box\, (Q \,{}_p\!\oplus R) & = & (P \,\Box\, Q) \,{}_p\!\oplus (P \,\Box\, R) \\
(\mathbf{D2}) & a.P \,\Box\, (Q \sqcap R) & = & (a.P \,\Box\, Q) \sqcap (a.P \,\Box\, R) \\
(\mathbf{D3}) & (P_1 \sqcap P_2) \,\Box\, (Q_1 \sqcap Q_2) & = & (P_1 \,\Box\, (Q_1 \sqcap Q_2)) \sqcap (P_2 \,\Box\, (Q_1 \sqcap Q_2)) \\
 & & & \quad \sqcap ((P_1 \sqcap P_2) \,\Box\, Q_1) \sqcap ((P_1 \sqcap P_2) \,\Box\, Q_2)
\end{array}
$$

Figure 5.8: Common equations

In order to focus on the essentials we consider just those pCSP processes that do not use the parallel operator $|_A$; we call the resulting sub-language nCSP. For a brief discussion of the axiomatisation for terms involving $|_A$ and the other parallel operators commonly used in CSP see Section 5.14.

Let us write $P =_E Q$ for equivalences that can be derived using the equations given in Figure 5.8. Given the way we defined the syntax of pCSP, axiom (**D1**) is merely a case of abbreviation-expansion; thanks to (**D1**) there is no need for (meta-)variables ranging over the sub-sort of state-based processes anywhere in the axioms. Many of the standard equations for CSP [Hoa85b] are missing; they are not sound for $\simeq_{FS}$. Typical examples include:

$$
\begin{aligned}
a.(P \sqcap Q) &= a.P \sqcap a.Q \\
P &= P \,\Box\, P \\
P \,\Box\, (Q \sqcap R) &= (P \,\Box\, Q) \sqcap (P \,\Box\, R) \\
P \sqcap (Q \,\Box\, R) &= (P \sqcap Q) \,\Box\, (P \sqcap R)
\end{aligned}
$$

For a detailed discussion of the standard equations for CSP in the presence of probabilistic processes see Section 4 of [DvGH$^+$07a].

Proposition 5.55 *Suppose $P =_E Q$. Then $P \simeq_{FS} Q$.*

Proof: Because of Theorem 5.26, that $\sqsubseteq_{FS}$ is a precongruence, it is sufficient to exhibit witness failure simulations for the axioms in Figure 5.8. These are exactly the same as the witness simulations for the same axioms, given in [DvGH$^+$07a]. The only axiom for which it is nontrivial to check that these simulations are in fact failure simulations is (**EI**). That axiom, as stated in [DvGH$^+$07a], is unsound here; it will return in the next section as (**May0**). But the special case of $a = b$ yields the axiom (**EI**) above, and then the witness simulation from [DvGH$^+$07a] is a failure simulation indeed. □

As $\simeq_S$ is a less discriminating equivalence than $\simeq_{FS}$ it follows that $P =_E Q$ implies $P \simeq_S Q$.

This equational theory allows us to reduce terms to a form in which the external choice operator is applied to prefix terms only.

Definition 5.56 (Normal forms) *The set of* normal forms *N is given by the following grammar:*

$$
N ::= N_1 \,{}_p\!\oplus N_2 \;\mid\; N_1 \sqcap N_2 \;\mid\; \bigsqcap_{i \in I} a_i.N_i
$$

Proposition 5.57 *For every $P \in$ nCSP there is a normal form N such that $P =_E N$.*

May:

$$\begin{array}{lrcl}
(\mathbf{May0}) & a.P \,\Box\, b.Q & = & a.P \sqcap b.Q \\
(\mathbf{May1}) & P & \sqsubseteq & P \sqcap Q \\
(\mathbf{May2}) & \mathbf{0} & \sqsubseteq & P \\
(\mathbf{May3}) & a.(P \,{}_p\!\oplus\, Q) & \sqsubseteq & a.P \,{}_p\!\oplus\, a.Q
\end{array}$$

Must:

$$\begin{array}{lrcl}
(\mathbf{Must1}) & P \sqcap Q & \sqsubseteq & Q \\
(\mathbf{Must2}) & R \sqcap \bigsqcap_{i\in I} \bigoplus_{j\in J_i} p_j \cdot (a_i.Q_{ij} \,\Box\, P_{ij}) & \sqsubseteq & \bigsqcap_{i\in I} a_i. \bigoplus_{j\in J_i} p_j \cdot Q_{ij}, \\
& \text{provided} & & \mathit{inits}(R) \subseteq \{a_i\}_{i\in I}
\end{array}$$

Figure 5.9: Inequations

Proof: A fairly straightforward induction, heavily relying on (**D1**)–(**D3**). □

We can also show that the axioms (**P1**)–(**P3**) and (**D1**) are in some sense all that are required to reason about probabilistic choice. Let $P =_{\text{prob}} Q$ denote that equivalence of P and Q can be derived using those axioms alone. Then we have the following property.

Lemma 5.58 *Let $P, Q \in \mathsf{nCSP}$. Then $[\![P]\!] = [\![Q]\!]$ implies $P =_{prob} Q$.*

Here $[\![P]\!] = [\![Q]\!]$ says that $[\![P]\!]$ and $[\![Q]\!]$ are the very same distributions of state-based processes in sCSP; this is a much stronger prerequisite than P and Q being testing equivalent.
Proof: The axioms (**P1**)–(**P3**) and (**D1**) essentially allow any processes to be written in the unique form $\bigoplus_{i\in I} p_i s_i$, where the $s_i \in \mathsf{sCSP}$ are all different. □

5.12 Inequational theories

In order to characterise the simulation preorders, and the associated testing preorders, we introduce *inequations*. We write $P \sqsubseteq_{E_{\text{may}}} Q$ when $P \sqsubseteq Q$ is derivable from the inequational theory obtained by adding the four *may* inequations in Figure 5.9 to the equations in Figure 5.8. The first three additions, (**May0**)–(**May2**), are used in the standard testing theory of CSP [Hoa85b, DNH84, Hen88]. For the *must* case, in addition to the standard inequation (**Must1**), we require an inequational schema, (**Must2**); this uses the notation $\mathit{inits}(P)$ to denote the (finite) set of initial actions of P. Formally,

$$\begin{array}{rcl}
\mathit{inits}(0) & = & \emptyset \\
\mathit{inits}(a.P) & = & \{a\} \\
\mathit{inits}(P \,{}_p\!\oplus\, Q) & = & \mathit{inits}(P) \cup \mathit{inits}(Q) \\
\mathit{inits}(P \,\Box\, Q) & = & \mathit{inits}(P) \cup \mathit{inits}(Q) \\
\mathit{inits}(P \sqcap Q) & = & \{\tau\}
\end{array}$$

The axiom (**Must2**) can equivalently be formulated as follows:

$$\bigoplus_{k\in K} \Box_{\ell\in L_k} a_{k\ell}.R_{k\ell} \sqcap \bigsqcap_{i\in I} \bigoplus_{j\in J_i} p_j \cdot (a_i.Q_{ij} \,\Box\, P_{ij}) \;\sqsubseteq\; \bigsqcap_{i\in I} a_i. \bigoplus_{j\in J_i} p_j \cdot Q_{ij},$$

$$\text{provided} \quad \{a_{k\ell} \mid k \in K, \ell \in K_k\} \subseteq \{a_i \mid i \in I\}\,.$$

This is the case because a term R satisfies $\mathit{inits}(R) \subseteq \{a_i\}_{i\in I}$ iff it can be converted into the form $\bigoplus_{k\in K} \Box_{\ell\in L_k} a_{k\ell}.R_{k\ell}$ by means of axioms (**D1**), (**P1**)–(**P3**) and (**E1**)–(**E3**) of Figure 5.9. This axiom can also be reformulated in an equivalent but more semantic style:

$$\begin{array}{lrcl}
(\mathbf{Must2'}) & R \sqcap \bigsqcap_{i\in I} P_i & \sqsubseteq & \Box_{i\in I}\, a_i.Q_i, \\
\text{provided} & [\![P_i]\!] \xrightarrow{a_i} [\![Q_i]\!] \text{ and } [\![R]\!] \overset{X}{\not\longrightarrow} \text{ with } X = Act \backslash \{a_i\}_{i\in I}. & &
\end{array}$$

This is the case because $[\![P]\!] \xrightarrow{a} [\![Q]\!]$ iff, up to the axioms in Figure 5.8, P has the form $\bigoplus_{j\in J} p_j \cdot (a.Q_j \,\Box\, P_j)$ and Q has the form $a.\bigoplus_{j\in J} p_j \cdot Q_j$ for certain P_j, Q_j and p_j, for $j \in J$.

Note that (**Must2**) can be used, together with (**I1**), to derive the dual of (**May3**) via the following inference:

$$\begin{array}{lll} a.P \,{}_p\!\oplus\, a.Q & =_E & (a.P \,{}_p\!\oplus\, a.Q) \sqcap (a.P \,{}_p\!\oplus\, a.Q) \\ & \sqsubseteq_{E_{\mathrm{must}}} & a.(P \,{}_p\!\oplus\, Q) \end{array}$$

where we write $P \sqsubseteq_{E_{\mathrm{must}}} Q$ when $P \sqsubseteq Q$ is derivable from the resulting inequational theory.

An important inequation that follows from (**May1**) and (**P1**) is

$$(\mathbf{May4}) \quad P \,{}_p\!\oplus\, Q \;\sqsubseteq_{E_{\mathrm{may}}}\; P \sqcap Q$$

saying that any probabilistic choice can be simulated by an internal choice. It is derived as follows:

$$\begin{array}{lll} P \,{}_p\!\oplus\, Q & \sqsubseteq_{E_{\mathrm{may}}} & (P \sqcap Q) \,{}_p\!\oplus\, (P \sqcap Q) \\ & =_E & (P \sqcap Q) \end{array}$$

Likewise, we have

$$P \sqcap Q \;\sqsubseteq_{E_{\mathrm{must}}}\; P \,{}_p\!\oplus\, Q\,.$$

Theorem 5.59 *For P, Q in nCSP, it holds that*

(i) $P \sqsubseteq_S Q$ if and only if $P \sqsubseteq_{E_{may}} Q$

(ii) $P \sqsubseteq_{FS} Q$ if and only if $P \sqsubseteq_{E_{must}} Q$.

Proof: For one direction it is sufficient to check that the inequations, and the inequational schema in Figure 5.9 are sound. For $\sqsubseteq_S$ this has been done in [DvGH$^+$07a], and the soundness of (**Must1**) and (**Must2′**) for $\sqsubseteq_{FS}$ is trivial. The converse, completeness, is established in the next section. □

5.13 Completeness

The completeness proof of Theorem 5.59 depends on the following variation on the *Derivative lemma* of [Mil89a]:

Lemma 5.60 (Derivative lemma) *Let $P, Q \in \mathsf{nCSP}$.*

(i) If $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![Q]\!]$ then $P \sqsubseteq_{E_{must}} Q$ and $Q \sqsubseteq_{E_{may}} P$.

(ii) If $[\![P]\!] \stackrel{a}{\Longrightarrow} [\![Q]\!]$ then $a.Q \sqsubseteq_{E_{may}} P$.

Proof: The proof of (i) proceeds in four stages. We only deal with $\sqsubseteq_{E_{\mathrm{may}}}$, as the proof for $\sqsubseteq_{E_{\mathrm{must}}}$ is entirely analogous.

First we show by structural induction on $s \in \mathsf{sCSP} \cap \mathsf{nCSP}$ that $s \xrightarrow{\tau} [\![Q]\!]$ implies $Q \sqsubseteq_{E_{\mathrm{may}}} s$. So suppose $s \xrightarrow{\tau} [\![Q]\!]$. In case s has the form $P_1 \sqcap P_2$ it follows by the operational semantics of pCSP that $Q = P_1$ or $Q = P_2$. Hence $Q \sqsubseteq_{E_{\mathrm{may}}} s$ by (**May1**). The only other possibility is that s has the form $s_1 \,\Box\, s_2$. In that case there must be a distribution Δ such that either $s_1 \xrightarrow{\tau} \Delta$ and $[\![Q]\!] = \Delta \,\Box\, s_2$, or $s_2 \xrightarrow{\tau} \Delta$ and $[\![Q]\!] = s_1 \,\Box\, \Delta$. Using symmetry, we may restrict attention to the first case. Let R be a term such that $[\![R]\!] = \Delta$. Then $[\![R \,\Box\, s_2]\!] = \Delta \,\Box\, s_2 = [\![Q]\!]$, so Lemma 5.58 yields $Q =_{\mathrm{prob}} R \,\Box\, s_2$. By induction we have $R \sqsubseteq_{E_{\mathrm{may}}} s_1$, hence $R \,\Box\, s_2 \sqsubseteq_{E_{\mathrm{may}}} s_1 \,\Box\, s_2$, and thus $Q \sqsubseteq_{E_{\mathrm{may}}} s$.

Now we show that $s \xrightarrow{\hat{\tau}} [\![Q]\!]$ implies $Q \sqsubseteq_{E_{\mathrm{may}}} s$. This follows because $s \xrightarrow{\hat{\tau}} [\![Q]\!]$ means that either $s \xrightarrow{\tau} [\![Q]\!]$ or $[\![Q]\!] = \overline{s}$, and in the latter case Lemma 5.58 yields $Q =_{\mathrm{prob}} s$.

Next we show that $[\![P]\!] \xrightarrow{\hat{\tau}} [\![Q]\!]$ implies $Q \sqsubseteq_{E_{\mathrm{may}}} P$. So suppose $[\![P]\!] \xrightarrow{\hat{\tau}} [\![Q]\!]$, that is

$$[\![P]\!] = \sum_{i\in I} p_i \cdot \overline{s_i} \qquad s_i \xrightarrow{\hat{\tau}} [\![Q_i]\!] \qquad [\![Q]\!] = \sum_{i\in I} p_i \cdot [\![Q_i]\!]$$

for some I, $p_i \in (0,1]$, $s_i \in \mathsf{sCSP} \cap \mathsf{nCSP}$ and $Q_i \in \mathsf{nCSP}$. Now

1. $[\![P]\!] = [\![\bigoplus_{i\in I} p_i \cdot s_i]\!]$. By Lemma 5.58 we have $P =_{\rm prob} \bigoplus_{i\in I} p_i \cdot s_i$.
2. $[\![Q]\!] = [\![\bigoplus_{i\in I} p_i \cdot Q_i]\!]$. Again Lemma 5.58 yields $Q =_{\rm prob} \bigoplus_{i\in I} p_i \cdot Q_i$.
3. $s_i \xrightarrow{\hat{\tau}} [\![Q_i]\!]$ implies $Q_i \sqsubseteq_{E_{\rm may}} s_i$. Therefore, $\bigoplus_{i\in I} p_i \cdot Q_i \sqsubseteq_{E_{\rm may}} \bigoplus_{i\in I} p_i \cdot s_i$.

Combining (1), (2) and (3) we obtain $Q \sqsubseteq_{E_{\rm may}} P$.

Finally, the general case, when $[\![P]\!] \xrightarrow{\hat{\tau}}{}^* \Delta$, is now a simple inductive argument on the length of the derivation.

The proof of (ii) is similar: first we treat the case when $s \xrightarrow{a} [\![Q]\!]$ by structural induction, using (**May2**); then the case $[\![P]\!] \xrightarrow{a} [\![Q]\!]$, exactly as above; and finally use part (i) to derive the general case. □

The completeness result now follows from the following two propositions.

Proposition 5.61 *Let P and Q be in* nCSP. *Then $P \sqsubseteq_S Q$ implies $P \sqsubseteq_{E_{may}} Q$.*

Proof: The proof is by structural induction on P and Q, and we may assume that both P and Q are in normal form because of Proposition 5.57. So take $P, Q \in$ pCSP and suppose the claim has been established for all subterms P' of P and Q' of Q, of which at least one of the two is a strict subterm. We start by proving that if $P \in$ sCSP then we have

$$P \lhd_S [\![Q]\!] \quad \text{implies} \quad P \sqsubseteq_{E_{\rm may}} Q. \tag{5.27}$$

There are two cases to consider.

1. P has the form $P_1 \sqcap P_2$. Since $P_i \sqsubseteq_{E_{\rm may}} P$ we know $P_i \sqsubseteq_S P \sqsubseteq_S Q$. We use induction to obtain $P_i \sqsubseteq_{E_{\rm may}} Q$, from which the result follows using (**I1**).
2. P has the form $\bigsqcap_{i\in I} a_i.P_i$. If I contains two or more elements then P may also be written as $\bigsqcap_{i\in I} a_i.P_i$, using (**May0**) and (**D2**), and we may proceed as in case (1) above. If I is empty, that is P is $\mathbf{0}$, then we can use (**May2**). So we are left with the possibility that P is $a.P'$. Thus suppose that $a.P' \lhd_S [\![Q]\!]$. We proceed by a case analysis on the structure of Q.
   - Q is $a.Q'$. We know from $a.P' \lhd_S [\![a.Q']\!]$ that $[\![P']\!] \overline{\lhd_S} \Theta$ for some Θ with $[\![Q']\!] \xRightarrow{\hat{\tau}} \Theta$, thus $P' \sqsubseteq_S Q'$. Therefore, we have $P' \sqsubseteq_{E_{\rm may}} Q'$ by induction. It follows that $a.P' \sqsubseteq_{E_{\rm may}} a.Q'$.
   - Q is $\bigsqcap_{j\in I} a_j.Q_j$ with at least two elements in J. We use (**May0**) and then proceed as in the next case.
   - Q is $Q_1 \sqcap Q_2$. We know from $a.P' \lhd_S [\![Q_1 \sqcap Q_2]\!]$ that $[\![P']\!] \overline{\lhd_S} \Theta$ for some Θ such that one of the following two conditions holds
     (a) $[\![Q_i]\!] \xRightarrow{a} \Theta$ for $i = 1$ or 2. In this case, $a.P' \lhd_S [\![Q_i]\!]$, hence $a.P' \sqsubseteq_S Q_i$. By induction we have $a.P' \sqsubseteq_{E_{\rm may}} Q_i$; then we apply (**May1**).
     (b) $[\![Q_1]\!] \xRightarrow{a} \Theta_1$ and $[\![Q_2]\!] \xRightarrow{a} \Theta_2$ such that $\Theta = p \cdot \Theta_1 + (1-p) \cdot \Theta_2$ for some $p \in (0,1)$. Let $\Theta_i = [\![Q_i']\!]$ for $i = 1, 2$. By the Derivative Lemma, we have $a.Q_1' \sqsubseteq_{E_{\rm may}} Q_1$ and $a.Q_2' \sqsubseteq_{E_{\rm may}} Q_2$. Clearly, $[\![Q_1' \,{}_p\!\oplus Q_2']\!] = \Theta$, thus $P' \sqsubseteq_S Q_1' \,{}_p\!\oplus Q_2'$. By induction, we infer that $P' \sqsubseteq_{E_{\rm may}} Q_1' \,{}_p\!\oplus Q_2'$. So

$$\begin{array}{lll} a.P' & \sqsubseteq_{E_{\rm may}} & a.(Q_1' \,{}_p\!\oplus Q_2') \\ & \sqsubseteq_{E_{\rm may}} & a.Q_1' \,{}_p\!\oplus a.Q_2' \qquad (\textbf{May3}) \\ & \sqsubseteq_{E_{\rm may}} & Q_1 \,{}_p\!\oplus Q_2 \\ & \sqsubseteq_{E_{\rm may}} & Q_1 \sqcap Q_2 \qquad (\textbf{May4}) \end{array}$$

   - Q is $Q_1 \,{}_p\!\oplus Q_2$. We know from $a.P' \lhd_S [\![Q_1 \,{}_p\!\oplus Q_2]\!]$ that $[\![P']\!] \overline{\lhd_S} \Theta$ for some Θ such that $[\![Q_1 \,{}_p\!\oplus Q_2]\!] \xRightarrow{a} \Theta$. From Lemma 5.19 we know that Θ must take the form $p \cdot [\![Q_1']\!] + (1-p) \cdot [\![Q_2']\!]$, where $[\![Q_i]\!] \xRightarrow{a} [\![Q_i']\!]$ for $i = 1, 2$. Hence $P' \sqsubseteq_S Q_1' \,{}_p\!\oplus Q_2'$, and by induction we get $P' \sqsubseteq_{E_{\rm may}} Q_1' \,{}_p\!\oplus Q_2'$. Then we can derive $a.P' \sqsubseteq_{E_{\rm may}} Q_1 \,{}_p\!\oplus Q_2$ as in the previous case.

Now we use (5.27) to show that $P \sqsubseteq_S Q$ implies $P \sqsubseteq_{E_{\text{may}}} Q$. Suppose $P \sqsubseteq_S Q$. Applying Definition 5.15 with the understanding that any distribution $\Theta \in \mathcal{D}(\mathsf{sCSP})$ can be written as $[\![Q']\!]$ for some $Q' \in \mathsf{pCSP}$, this means that $[\![P]\!] \ \overline{\lhd_S} \ [\![Q']\!]$ for some $[\![Q]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![Q']\!]$. The Derivative Lemma yields $Q' \sqsubseteq_{E_{\text{may}}} Q$. So it suffices to show $P \sqsubseteq_{E_{\text{may}}} Q'$. We know that $[\![P]\!] \ \overline{\lhd_S} \ [\![Q']\!]$ means that

$$[\![P]\!] = \sum_{k\in K} r_k \cdot \overline{t_k} \qquad t_k \lhd_S [\![Q'_k]\!] \qquad [\![Q']\!] = \sum_{k\in K} r_k \cdot [\![Q'_k]\!]$$

for some K, $r_k \in (0,1]$, $t_k \in \mathsf{sCSP}$ and $Q'_k \in \mathsf{pCSP}$. Now

1. $[\![P]\!] = [\![\bigoplus_{k\in K} r_k \cdot t_k]\!]$. By Lemma 5.58 we have $P =_{\text{prob}} \bigoplus_{k\in K} r_k \cdot t_k$.
2. $[\![Q']\!] = [\![\bigoplus_{k\in K} r_k \cdot Q'_k]\!]$. Again Lemma 5.58 yields $Q' =_{\text{prob}} \bigoplus_{k\in K} r_k \cdot Q'_k$.
3. $t_k \lhd_S [\![Q'_k]\!]$ implies $t_k \sqsubseteq_{E_{\text{may}}} Q'_k$ by (5.27). Therefore, $\bigoplus_{k\in K} r_k \cdot t_k \sqsubseteq_{E_{\text{may}}} \bigoplus_{k\in K} r_k \cdot Q'_k$.

Combining (1), (2) and (3) we obtain $P \sqsubseteq_{E_{\text{may}}} Q'$, hence $P \sqsubseteq_{E_{\text{may}}} Q$. □

Proposition 5.62 *Let P and Q be in nCSP. Then $P \sqsubseteq_{FS} Q$ implies $P \sqsubseteq_{E_{must}} Q$.*

Proof: Similar to the proof of Proposition 5.61, but using a reversed orientation of the preorders. The only real difference is the case (2), which we consider now. So assume $Q \lhd_{FS} [\![P]\!]$, where Q has the form $\Box_{i\in I} a_i.Q_i$. Let X be any set of actions such that $X \cap \{a_i\}_{i\in I} = \emptyset$; then $\Box_{i\in I} a_i.Q_i \stackrel{X}{\not\longrightarrow}$. Therefore, there exists a P' such that $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![P']\!] \stackrel{X}{\not\longrightarrow}$. By the Derivative lemma,

$$P \sqsubseteq_{E_{\text{must}}} P' \tag{5.28}$$

Since $\Box_{i\in I} a_i.Q_i \stackrel{a_i}{\longrightarrow} [\![Q_i]\!]$, there exist P_i, P'_i, P''_i such that $[\![P]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![Pi]\!] \stackrel{a_i}{\longrightarrow} [\![P'_i]\!] \stackrel{\hat{\tau}}{\Longrightarrow} [\![P''_i]\!]$ and $[\![Q_i]\!] \ \overline{\lhd_{FS}} \ [\![P''_i]\!]$. Now

$$P \sqsubseteq_{E_{\text{must}}} P_i \tag{5.29}$$

using the Derivative lemma, and $P'_i \sqsubseteq_{FS} Q_i$, by Definition 5.15. By induction, we have $P'_i \sqsubseteq_{E_{\text{must}}} Q_i$, hence

$$\Box_{i\in I} a_i.P'_i \sqsubseteq_{E_{\text{must}}} \Box_{i\in I} a_i.Q_i \tag{5.30}$$

The desired result is now obtained as follows:

$$\begin{array}{lll} P & \sqsubseteq_{E_{\text{must}}} & P' \sqcap \prod_{i\in I} P_i \quad \text{by (I1), (5.28) and (5.29)} \\ & \sqsubseteq_{E_{\text{must}}} & \Box_{i\in I} a_i.P'_i \quad \text{by (Must2}') \\ & \sqsubseteq_{E_{\text{must}}} & \Box_{i\in I} a_i.Q_i \quad \text{by (5.30)} \end{array}$$ □

Propositions 5.61 and 5.62 give us the completeness result stated in Theorem 5.59.

5.14 Summary and discussions

In this chapter we have studied three different aspects of may- and must testing preorders for finite processes: (i) we have shown that the may preorder can be characterised as a co-inductive simulation relation, and the must preorder as a failure simulation relation; (ii) we have given a characterisation of both preorders in a finitary modal logic; and (iii) we have also provided complete axiomatisations for both preorders over a probabilistic version of recursion-free CSP. Although we omitted our parallel operator $|_A$ from the axiomatisations, it and similar CSP and CCS-like parallel operators can be handled using standard techniques, in the must case at the expense of introducing auxiliary operators.

5.14.1 Probabilistic models

Models for probabilistic concurrent systems have been studied for a long time [Rab63, Der70, Var85b, JP89]. One of the first models obtained as a simple adaptation of the traditional labelled transition systems from concurrency theory appears in [LS91]. Their *probabilistic transition systems* are classical labelled transition systems, where in addition every transition is labelled with a probability, a real number in the interval [0,1], such that for every state s and every action a, the probabilities of all a-labelled transitions leaving s sum up to either 0 or 1.

In [GJS90] a similar model was proposed, but where the probabilities of *all* transitions leaving s sum up to either 0 or 1. [GSST90] propose the terminology *reactive* for the type of model studied in [LS91], and *generative* for the type of model studied in [GJS90]. In a generative model, a process can be considered to spontaneously generate actions, unless restricted by the environment; in generating actions, a probabilistic choice is made between all transitions that can be taken from a given state, even if they have different labels. In a reactive model, on the other hand, processes are supposed to perform actions only in response to requests by the environment. The choice between two different actions is therefore not under the control of the process itself. When the environment requests a specific action, a probabilistic choice is made between all transitions (if any) that are labelled with the requested action.

In the above-mentioned models, the nondeterministic choice that can be modelled by non-probabilistic labelled transition systems is *replaced* by a probabilistic choice (and in the generative model also a deterministic choice, a choice between different actions, is made probabilistic). Hence reactive and generative probabilistic transition systems do not generalise non-probabilistic labelled transition systems. A model, or rather a calculus, that features both nondeterministic and reactive probabilistic choice was proposed in [HJ90]. It was slightly reformulated in [SL94] under the name *simple probabilistic automata*, and is essentially the same model we use in this paper.

Following the classification above, our model is reactive rather than generative. The reactive model of [LS91] can be reformulated by saying that a state s has at most one outgoing transition for any action a, and this transition ends in a probability distribution over its successor states. The generalisation of [SL94], that we use here as well, is that a state can have multiple outgoing transitions with the same label, each ending in a probability distribution. Simple probabilistic automata are a special case of the *probabilistic automata* of [SL94], that also generalise the generative models of probabilistic processes to a setting with nondeterministic choice.

5.14.2 Bisimulation, and the alternating approach

Whereas the testing semantics explored in the present paper is based on the idea that processes should be *distinguished* only when there is a compelling reason to do so, (strong) bisimulation semantics [Mil89a] is based on the idea that processes should be *identified* only when there is a compelling reason to do so. It has been extended to reactive probabilistic processes in [LS91], to generative ones in [GSST90], and to processes combining nondeterminism and probability in [HJ90]. The latter paper also features a complete axiomatisation of a probabilistic extension of recursion-free CCS.

Weak and branching bisimulation [Mil89a, GW96] are versions of strong bisimulation that respect the hidden nature of the internal action τ. Generalisations of these notions to nondeterministic probabilistic processes appear, amongst others, in [SL94, Seg95, PLS00, AB01, BS01, DP05a, AW06], with complete axiomatisations reported in [BS01, DP05a, DPP05, ABW06]. The authors of these paper tend to distinguish whether they work in an *alternating* [PLS00, AB01, AW06, ABW06] or a *non-alternating* model of probabilistic processes [SL94, Seg95, DP05a, DPP05], the two approaches being compared in [BS01]. The non-alternating model stems from [SL94] and is similar to our model of Section 5.2.2. The alternating model is attributed to [HJ90], and resembles our graphical representation of processes in Section 5.2.4. It is easy to see that mathematically the alternating and non-alternating model can be translated into each other without loss of information [BS01]. The difference between the two is one of interpretation. In the alternating interpretation, the nodes of form $\circ$ in our graphical representations are interpreted as actual states a process can be in, whereas in the non-alternating representation they are not. Take for example the process $R_1 = a.(b \,{}_{\frac{1}{2}}\!\oplus\, c)$ depicted

in Figure 5.5. In the alternating representation this process passes through a state in which a has already happened, but the probabilistic choice between b and c has not yet been made. In the non-alternating interpretation on the other hand the execution of a is what constitutes this probabilistic choice; after doing a there is a fifty-fifty change of ending up in either state. Although in strong bisimulation semantics the alternating and non-alternating interpretation lead to the same semantic equivalence, in weak and branching bisimulation semantics the resulting equivalences are different, as illustrated in [PLS00, BS01, AW06]. Our testing and simulation preorders as presented here can be classified as non-alternating; however, we believe that an alternating approach would lead to the very same preorders.

Early additions of probability to CSP include work by Lowe [Low93], Seidel [Sei95] and Morgan et al. [MMSS96]; but all of them were forced to make compromises of some kind in order to address the potentially complicated interactions between the three forms of choice. The last [MMSS96] for example applied the Jones/Plotkin probabilistic powerdomain [JP89] directly to the failures model of CSP [BHR84], the resulting compromise being that probability distributed outwards through all other operators; one controversial result of that was that internal choice was no longer idempotent, and that it was "clairvoyant" in the sense that it could adapt to probabilistic-choice outcomes that had not yet occurred. Mislove addressed this problem in [Mis00] by presenting a denotational model in which internal choice distributed outwards through probabilistic choice. However, the distributivities of both [MMSS96] and [Mis00] constitute identifications that cannot be justified by our testing approach; see [DvGH$^+$07a].

In Jou and Smolka [JS90], as in [Low93, Sei95], probabilistic equivalences based on traces, failures and readies are defined. These equivalences are coarser than $\simeq_{\text{pmay}}$. For let

$$P := a.((b.d \,\Box\, c.e) \,{}_{\frac{1}{2}}\!\oplus\, (b.f \,\Box\, c.g))$$
$$Q := a.((b.d \,\Box\, c.g) \,{}_{\frac{1}{2}}\!\oplus\, (b.f \,\Box\, c.e)).$$

For example, the two processes cannot be distinguished by the equivalences of [JS90, Low93, Sei95]. However, we can tell them apart by the test:

$$T := a.((b.d.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.e.\omega) \sqcap (b.f.\omega \,{}_{\frac{1}{2}}\!\oplus\, c.g.\omega))$$

since $\mathcal{A}pply(T, P) = \{0, \frac{1}{2}, 1\}$ and $\mathcal{A}pply(T, Q) = \{\frac{1}{2}\}$, that is, $P \not\sqsubseteq_{\text{pmay}} Q$.

5.14.3 Testing

Probabilistic extensions of testing equivalences [DNH84] have been widely studied. There are two different proposals on how to include probabilistic choice: (i) a test should be non-probabilistic, i.e., there is no occurrence of probabilistic choice in a test [LS91, Chr90, JHSY94, KN98, GRN99]; or (ii) a test can be probabilistic, i.e., probabilistic choice may occur in tests as well as processes [CDSY99, YL92, Núñ03, JY95, Seg96, JY02, CCR$^+$03a]. This chatper adopts the second approach.

Some work [LS91, Chr90, CDSY99, Núñ03] does not consider nondeterminism but deals exclusively with *fully probabilistic* processes. In this setting a process passes a test with a unique probability instead of a set of probabilities, and testing preorders in the style of [DNH84] have been characterised in terms of *probabilistic traces* [CDSY99] and *probabilistic acceptance trees* [Núñ03]. Cazorla et al. [CCR$^+$03a] extended the results of [Núñ03] with nondeterminism, but suffered from the same problems as [MMSS96].

Generalisations of the testing theory of [DNH84] to probabilistic systems first appear in [Chr90] and [CSZ92], for generative processes without nondeterministic choice. The application of testing to the probabilistic processes we consider here stems from [YL92]. In [Seg96] a richer testing framework is proposed, for essentially the same class of processes, namely one in which multiple success actions ω_i for $i = 1, 2, \ldots$ exists, and the application of a test to a process yields not a set of real numbers, indicating success probabilities, but a set of tuples of real numbers, the i^{th} component in the tuple indicating the success probability of ω_i. Segala [Seg96] defined two preorders called trace distribution precongruence ($\sqsubseteq_{TD}$) and failure distribution precongruence ($\sqsubseteq_{FD}$). He proved that the former

coincides with an infinitary version of $\widehat{\sqsubseteq}^{\Omega}_{\text{pmay}}$ (cf. Definition 5.35) and that the latter coincides with an infinitary version of $\widehat{\sqsubseteq}^{\Omega}_{\text{pmust}}$. In [LSV03] it has been shown that $\sqsubseteq_{TD}$ coincides with a notion of simulation akin to $\sqsubseteq_S$. Other probabilistic extensions of simulation occurring in the literature are reviewed in [DvGH$^+$07a].

In [JHSY94], a testing theory is proposed that associates a *reward*, a non-negative real number, to every success-state in a test process; in calculating the set of results of applying a test to a process, the probabilities of reaching a success-state are multiplied by the reward associated to that state. They allow non-probabilistic tests only, but apply these to arbitrary nondeterministic probabilistic processes, and provide a trace-like denotational characterisation of the resulting may-testing preorder. Denotational characterisations of the variant of our testing preorders in which only τ-free processes are allowed as test-processes appear in [JY95, JY99]. These characterisations are improved in [JY02], discussed below.

In [CCR$^+$03b] a testing theory for nondeterministic probabilistic processes is developed in which, as in [MMSS96], all probabilistic choices are resolved first. A consequence of this is that the idempotence of internal choice (our axiom (**I1**)) must be sacrificed. Some papers distill preorders for probabilistic processes by means of *testing scenarios* in which the premise that a test is itself a process is given up. These include [LS91, KN98] and [SV03].

5.14.4 Simulations

Four different notions of simulation for probabilistic processes occur in the literature, each a generalisation of the well know concept of simulation for nondeterministic processes [Par81]. The most straightforward generalisation [JL91] defines a simulation as a relation $\mathcal{R}$ between states, satisfying, for all s, t, μ, Θ,

$$\text{if } s\mathcal{R}t \text{ and } s \xrightarrow{\mu} \Theta \text{ then there is a } \Delta' \text{ with } t \xrightarrow{\mu} \Delta' \text{ and } \Theta\, \overline{\mathcal{R}}\, \Delta'.$$

This simulation induces a preorder that does not satisfy the law

$$a.(P \,{}_p\!\oplus Q) \sqsubseteq a.P \,\Box\, a.Q$$

which holds in probabilistic *may* testing semantics. The reason is that the process $a.P \,\Box\, a.Q$ can answer the initial a-move of $a.(P \,{}_p\!\oplus Q)$ by taking either the a-move to P, or the a-move to Q, but not by a probabilistic combination of the two. Such probabilistic combinations are allowed in the probabilistic simulation of [SL94], which induces a coarser preorder on processes, satisfying the above law. In our terminology it can be defined by changing the requirement above into

$$\text{if } s\mathcal{R}t \text{ and } s \xrightarrow{\mu} \Theta \text{ then there is a } \Delta' \text{ with } \overline{t} \xrightarrow{\mu} \Delta' \text{ and } \Theta\, \overline{\mathcal{R}}\, \Delta'.$$

A *weak* version of this probabilistic simulation, abstracting from the internal action τ, weakens this requirement into

$$\text{if } s\mathcal{R}t \text{ and } s \xrightarrow{\mu} \Theta \text{ then there is a } \Delta' \text{ with } \overline{t} \stackrel{\hat{\mu}}{\Longrightarrow} \Delta' \text{ and } \Theta\, \overline{\mathcal{R}}\, \Delta'.$$

Nevertheless, also this probabilistic simulation does not satisfy all the laws we have shown to hold for probabilistic may testing. In particular, it does not satisfy the law

$$a.(P \,{}_p\!\oplus Q) \sqsubseteq a.P \,{}_p\!\oplus a.Q.$$

Consider for instance the processes $R_1 = a.b.c.(d \,{}_{\frac{1}{2}}\!\oplus e)$ and $R_2 = a.(b.c.d \,{}_{\frac{1}{2}}\!\oplus b.c.e)$. The law (**Q1**) above, which holds for probabilistic *may* testing, would yield $R_1 \sqsubseteq R_2$. If we are to relate these processes via a probabilistic simulation à la [SL94], the state $c.(d \,{}_{\frac{1}{2}}\!\oplus e)$ of R_1, reachable after an a and a b-step, needs to be related to the *distribution* $(c.d \,{}_{\frac{1}{2}}\!\oplus c.e)$ of R_2, containing the two states $a.b$ and $a.c$. This relation cannot be obtained through lifting, as this would entail relating the single state $c.(d \,{}_{\frac{1}{2}}\!\oplus e)$ to each of the states $c.d$ and $c.e$. Such a relation would not be sound, because $c.(d \,{}_{\frac{1}{2}}\!\oplus e)$ is able to perform the sequence of actions ce half of the time, whereas the process $c.d$ cannot mimic this.

In [JY02], another notion of simulation is proposed, whose definition is too complicated to explain in a few sentences. They show for a class of probabilistic processes that do not contain τ-actions, that probabilistic *may* testing is captured exactly by their notion of simulation. Nevertheless, their notion of simulation makes strictly more identifications than ours. As an example, consider the processes $R_1 = a \stackrel{1}{\frac{}{2}}\oplus (b \,\Box\, c)$ and $R_3 = (a \,\Box\, b) \stackrel{1}{\frac{}{2}}\oplus (a \,\Box\, c)$ of Example 5.10, which also appear in Section 5 of [JY02]. There it is shown that $R_1 \sqsubseteq R_3$ holds in their semantics. However, in our framework we have $R_1 \not\sqsubseteq_{\mathrm{pmay}} R_3$, as demonstrated in Example 5.10. The difference can only be explained by the circumstance that in [JY02] processes, and hence also tests, may not have internal actions. So this example shows that tests with internal moves can distinguish more processes than tests without internal moves, even when applied to processes that have no internal moves themselves.

Our notion of simulation first appears in [Seg95], although the preorder $\sqsubseteq_S$ of Definition 5.15 is new. Segala has no expressions that denote distributions and consequently is only interested in the restriction of the simulation preorder to states (*automata* in his framework). It turns out that for states s and t (which in our framework are expressions in the set S_p) we have $s \sqsubseteq_S t$ iff $s \lhd_s \overline{t}$, so on their common domain of definition, the simulation preorder of [Seg95] agrees with ours. This notion of simulation is strictly more discriminating than the simulation of [JY02], and strictly less discriminating than the ones from [SL94] and [JL91].

Chapter 6

Other Probabilistic Models

In this chapter we give a brief account of a probabilistic model called *distcrete-time Markov chains* and a probabilistic temporal logic called *probabilistic computation tree logic* as well as a model checking algorithm. We then move to two other important models: Markov decision processes and stochastic games. The description closely follows [KNP07, KKNP08].

6.1 Probabilistic Kripke structures

So far we have studied various semantics for pLTSs which are probabilistic extensions of LTSs. In this chapter we will see some models that can be considered as probabilistic extensions of Kripke structures. To put it in a nutshell, Kripke structures are directed graphs whose nodes are labelled. So it is not surprising that they have intimate relation with LTSs which are essentially directed graphs whose edges are labelled.

We presuppose a set AP of *atomic propositions*.

Definition 6.1 *A* Kripke structure *is a triple* $(S, L, \rightarrow)$*, where*

1. S *is a set of states*
2. L *is a labelling function* $L : S \to 2^{AP}$
3. $\rightarrow \subseteq S \times S$ *is the transition relation.*

Bisimulation relation can be easily defined on the states of a Kripke structure.

Definition 6.2 *A binary relation* $\mathcal{R}$ *on the states of a Kripke structure is a* simulation *if whenever* $s_1 \ \mathcal{R} \ s_2$*:*

- $L(s_1) = L(s_2)$
- *for all* s_1' *with* $s_1 \longrightarrow s_1'$ *there exists some* s_2' *such that* $s_2 \longrightarrow s_2'$ *and* $s_1' \ \mathcal{R} \ s_2'$*.*

The relation $\mathcal{R}$ *is a bisimulation if both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are simulations. We write* $\prec$ *and* $\sim$ *for the largest simulation and bisimulation, respectively.*

Given a model $\mathcal{M}$, which can be a Kripke structure or an LTS, a transformation f consists of a function that takes any state s in $\mathcal{M}$ to a state $f(s)$ in $f(\mathcal{M})$.

Proposition 6.3
1. *A Kripke structure* $\mathcal{K}$ *can be transformed into an LTS* $f(\mathcal{K})$ *such that* $s \sim t$ *iff* $f(s) \sim f(t)$*.*
2. *An LTS* $\mathcal{L}$ *can be transformed into a Kripke structure* $f(\mathcal{L})$ *such that* $s \sim t$ *iff* $f(s) \sim f(t)$*.*

□

We will prove the probabilistic counterpart of the proposition in the end of this section.

Definition 6.4 *A* probabilistic Kripke structure *is a triple* $(S, L, \rightarrow)$*, where*

1. S *is a set of states*
2. L *is a labelling function* $L : S \rightarrow 2^{AP}$
3. $\rightarrow \subseteq S \times \mathcal{D}(S)$ *is the transition relation.*

Bisimulation relation can also be defined on the states of a probabilistic Kripke structure.

Definition 6.5 *A binary relation* $\mathcal{R}$ *on the states of a Kripke structure is a* simulation *if whenever* $s_1 \; \mathcal{R} \; s_2$*:*

- $L(s_1) = L(s_2)$
- *for all* s_1' *with* $s_1 \longrightarrow \Delta_1$ *there exists some* Δ_2 *such that* $s_2 \longrightarrow \Delta_2$ *and* $\Delta_1 \; \overline{\mathcal{R}} \; \Delta_2$*.*

The relation $\mathcal{R}$ *is a bisimulation if both* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are simulations. We write* $\prec_K$ *and* $\sim_K$ *for the largest simulation and bisimulation, respectively.*

Proposition 6.6 *1. A probabilistic Kripke structure* $\mathcal{K}$ *can be transformed into a pLTS* $f(\mathcal{K})$ *such that* $s \sim_K t$ *iff* $f(s) \sim f(t)$*.*

2. A pLTS $\mathcal{L}$ *can be transformed into a probabilistic Kripke structure* $f(\mathcal{L})$ *such that* $s \sim t$ *iff* $f(s) \sim_K f(t)$*.*

Proof:

1. Let $\mathcal{K} = (S, L, \rightarrow)$, we take $f(\mathcal{K})$ be the pLTS $(\hat{S} \cup \{\bot\}, AP, \rightarrow)$, where

   - $\hat{S}$ is the set $\{\hat{s} \mid s \in S\}$ and $f(s) = \hat{s}$ for all $s \in S$.
   - $s \rightarrow \Delta$ iff $\hat{s} \xrightarrow{L(s)} \hat{\Delta}$; $s \not\longrightarrow$ iff $\hat{s} \xrightarrow{L(s)} \overline{\bot}$.

   ($\Rightarrow$) Let $\mathcal{R} = \{(\hat{s}, \hat{t}) \mid s \sim_K t\} \cup \{(\bot, \bot)\}$. We show that $\mathcal{R}$ is a bisimulation. Suppose $(\hat{s}, \hat{t}) \in \mathcal{R}$ and $\hat{s} \xrightarrow{a} \Delta$. There are two possibilities: (i) $\Delta = \overline{\bot}$; (ii) $\Delta = \hat{\Delta_1}$. In (i) we have that $L(s) = a$ and $s \not\longrightarrow$. Since $s \sim_K t$, we know that $L(t) = a$ and $t \not\longrightarrow$. Therefore, $\hat{t} \xrightarrow{a} \overline{\bot}$ and $\overline{\bot} \; \overline{\mathcal{R}} \; \overline{\bot}$ trivially. In (ii) we have that $L(s) = a$ and $s \longrightarrow \Delta_1$. Since $s \sim_K t$, we know that $L(t) = a$ and $t \longrightarrow \Theta_1$ with $\Delta_1 \; \overline{\sim_K} \; \Theta_1$. It follows that $\hat{t} \xrightarrow{a} \hat{\Theta_1}$ and $\hat{\Delta_1} \; \overline{\mathcal{R}} \; \hat{\Theta_1}$.

   ($\Leftarrow$) Let $\mathcal{R} = \{(s, t) \mid \hat{s} \sim \hat{t}\}$. We show that $\mathcal{R}$ is a bisimulation. Suppose $(s, t) \in \mathcal{R}$. Let $L(s) = a$, then we have $\hat{s} \xrightarrow{a} \Delta$ for some distribution Δ. Since $\hat{s} \sim \hat{t}$, there must be a matching transition from $\hat{t}$ that is labelled by a. Hence, we see that $L(t) = a$. Suppose $s \longrightarrow \Delta_1$, then $\hat{s} \xrightarrow{a} \hat{\Delta_1}$. To match the transition there exists a transition $\hat{t} \xrightarrow{a} \Theta$ with $\hat{\Delta_1} \; \overline{\approx} \; \Theta$. Note that $\Theta \neq \overline{\bot}$ because all states in $\lceil \hat{\Delta_1} \rceil$ are in the form $\hat{s}$ for some $s \in S$ and $\hat{s} \xrightarrow{L(s)}$ that cannot be matched by $\bot$ which is a deadlock state. So $\Theta = \hat{\Theta_1}$ for some Θ_1 such that $t \longrightarrow \Theta_1$. It is easy to see from $\hat{\Delta_1} \; \overline{\approx} \; \hat{\Theta_1}$ that $\Delta_1 \; \overline{\mathcal{R}} \; \Theta_1$.

2. Let $\mathcal{L} = (S, A, \rightarrow)$, we take $f(\mathcal{L})$ be the probabilistic Kripke structure $(\hat{S} \cup S', L, \rightarrow)$, where

   - $\hat{S} = \{\hat{s} \mid s \in S\}$ and $S' = \{s_{(a,\Delta)} \mid s \xrightarrow{a} \Delta\}$.
   - every transition $s \xrightarrow{a} \Delta$ in $\mathcal{L}$ is replaced by two transitions $\hat{s} \longrightarrow \overline{s_{(a,\Delta)}}$ and $s_{(a,\Delta)} \longrightarrow \hat{\Delta}$.
   - $L(\hat{s}) = \bot$ for all $s \in S$ and $L(s_{(a,\Delta)}) = a$.

($\Rightarrow$) We construct the relation $\mathcal{R} = \{(\hat{s}, \hat{t}) \mid s \sim t\} \cup \{(s_{(a,\Delta)}, t_{(a,\Theta)}) \mid \Delta \overline{\sim} \Theta\}$ and show $\mathcal{R}$ is a bisimulation. Suppose $(s_{(a,\Delta)}, t_{(a,\Theta)}) \in \mathcal{R}$, we know that $L(s_{(a,\Delta)}) = a = L(t_{(a,\Theta)})$, $s_{(a,\Delta)} \longrightarrow \hat{\Delta}$, $t_{(a,\Theta)} \longrightarrow \hat{\Theta}$ and $\Delta \overline{\sim} \Theta$. The last condition means that $\hat{\Delta}\ \overline{\mathcal{R}}\ \hat{\Theta}$. Suppose $(\hat{s}, \hat{t}) \in \mathcal{R}$. Clearly, $L(\hat{s}) = \bot = L(\hat{t})$. If $\hat{s} \longrightarrow \overline{s_{(a,\Delta)}}$, then we must have $s \xrightarrow{a} \Delta$. Since $s \sim t$, there is a matching transition $t \xrightarrow{a} \Theta$ such that $\Delta \overline{\sim} \Theta$. It follows that $\hat{t} \longrightarrow \overline{s_{(a,\Theta)}}$ and $\overline{s_{(a,\Delta)}}\ \overline{\mathcal{R}}\ \overline{t_{(a,\Theta)}}$.

($\Leftarrow$) We show that the relation $\mathcal{R} = \{(s,t) \mid \hat{s} \sim_K \hat{t}\}$ is a bisimulation. Suppose $(s,t) \in \mathcal{R}$ and $s \xrightarrow{a} \Delta$. Then we know that $\hat{s} \longrightarrow \overline{s_{(a,\Delta)}}$ and $s_{(a,\Delta)} \longrightarrow \hat{\Delta}$. Since $\hat{s} \sim_K \hat{t}$, we have $\hat{t} \longrightarrow \Theta_1$ for some distribution Θ_1 such that $\overline{s_{(a,\Delta)}}\ \overline{\sim_K}\ \Theta_1$. Note that each state $\hat{t}$ only leads to point distributions after one step of transition in $f(\mathcal{L})$. Hence, $\Theta_1 = \overline{t_{(a,\Theta)}}$ for some distribution Θ and moreover $s_{(a,\Delta)} \sim_K t_{(a,\Theta)}$. Since $s_{(a,\Delta)}$ has the only transition $s_{(a,\Delta)} \longrightarrow \Delta$ and $t_{(a,\Theta)}$ has the only transition $t_{(a,\Theta)} \longrightarrow \Theta$, we infer the required results that $\Delta\ \overline{\sim_K}\ \Theta$ and $t \xrightarrow{a} \Theta$.

□

6.2 Discrete-time Markov chains

Unlike probabilistic Kripke structures, which *add* probailistic choices to Kripke structures, Markov chains *replace* nondeterministic choices in Kripke structures with probabilistic choices.

Definition 6.7 *A labelled* distcrete-time Markov chains *(DTMC) is a tuple* $(S, s^\circ, \boldsymbol{P}, L)$ *where*

- S *is a finite set of states;*
- s° *is the initial state;*
- $\boldsymbol{P} : S \times S \to [0,1]$ *is the* transition probability matrix *where* $\sum_{s' \in S} \boldsymbol{P}(s,s') = 1$ *for all* $s \in S$*;*
- $L : S \to 2^{AP}$ *is a labelling function which assigns to each state* $s \in S$ *the set* $L(s)$ *of atomic propositions that are valid in the state.*

Each element $\mathbf{P}(s,s')$ of the transition probability matrix gives the probability of making a transition from state s to state s'. The probabilities of transitions emanating from a single state must sum up to one. To satisfy this requirement, we model a terminating state, which cannot move to another state, by adding a self-loop with probability 1.

Example 6.8 *Consider the DTMC* $M = (\{s_0, s_1, s_2\}, s_0, \boldsymbol{P}, L)$*. It has three states with* s_0 *being the initial state. The transition probability matrix* $\boldsymbol{P}$ *is given by:*

$$\boldsymbol{P} = \begin{pmatrix} 0 & 0.1 & 0.9 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The atomic propositions used to label states are taken from the set $AP = \{fail, succ\}$*. The labelling function* L *allows us to assign meaningful names to states. It is defined as follows:*

$$L(s_0) = \emptyset, \quad L(s_1) = \{fail\}, \quad L(s_2) = \{succ\}.$$

In Figure 6.1 we gives the graphical presentation of the DTMC. States are drawn as circles and transitions as arrows, labelled with their associated probabilities. The initial state is indicated by an incoming short arrow. The DTMC models a process that tries to send a message. With probability 0.9 *it successfully sends the message, and with probability* 0.1 *it fails and then restarts the process.*

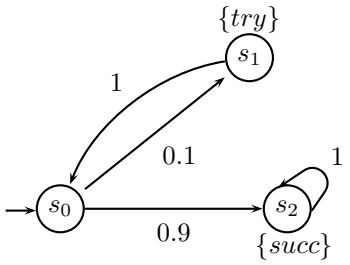


Figure 6.1: A graphical presentation of DTMC

An execution of a DTMC $M = (S, s^\circ, \mathbf{P}, L)$ is represented by a *path* obtained by resolving the probabilistic choice at each state. Formally, a path ξ is a non-empty sequence of states $s_0 s_1 s_2 ...$ where $s_i \in S$ and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. A path can be either finite or infinite. We write $\xi(i)$ for the i-th state of a path ξ, and $|\xi|$ the length of ξ which is the number of transitions gone through by the path. A finite path ξ_1 is said to be a *prefix* of the infinite path ξ_2 if $\xi_1(i) = \xi_2(i)$ for all $0 \leq i \leq |\xi_1|$. The set of all finite and infinite paths of M starting in state s are denoted by $Path_f^M(s)$ and $Path^M(s)$, respectively.

In order to reason about the probabilistic behaviour of the DTMC, we need to determine the probability that certain paths are taken. Below we define a probability measure by using the transition probability matrix $\mathbf{P}$. For any finite path $\xi \in Path_f^M(s)$, we define the probability

$$\mathbf{P}_s(\xi) := \begin{cases} 1 & \text{if } n = 0 \\ \mathbf{P}(\xi(0), \xi(1)) \cdots \mathbf{P}(\xi(n-1), \xi(n)) & \text{otherwise} \end{cases}$$

where $n = |\xi|$. We define the *cylinder set* or *cone* $C(\xi) \subseteq Path^M(s)$ as the set of all infinite paths with prefix ξ; formally

$$C(\xi) := \{\xi' \in Path^M(s) \mid \xi \text{ is a prefix of } \xi'\}.$$

By Proposition 1.23, there exists a unique smallest σ-algebra on $Path^M(s)$, denoted $\mathcal{X}_{Path^M(s)}$, which contains all the sets $C(\xi)$ with ξ ranging over the finite paths $Path_f^M(s)$. As the set of cylinders forms a semi-ring over $(Path^M(s), \mathcal{X}_{Path^M(s)})$, we can apply Theorem 1.26 and define Pr_s on $(Path^M(s), \mathcal{X}_{Path^M(s)})$ as the unique measure such that

$$Pr_s(C(\xi)) := \mathbf{P}_s(\xi) \text{ for all } \xi \in Path_f^M(s).$$

Note that $C(s) = Path^M(s)$ and $\mathbf{P}_s(s) = 1$, so Pr_s is in fact a probability measure.

Example 6.9 *Consider the DTMC in Figure 6.1. There are three distinct paths of length* 3 *starting in state* s_0*. The probabilities given to the cylinder sets associated with the paths by the probability measure* Pr_{s_0} *are:*

$$Pr_{s_0}(C(s_0 s_1 s_0 s_1)) = 0.1 \cdot 1.0 \cdot 0.1 = 0.01$$
$$Pr_{s_0}(C(s_0 s_1 s_0 s_2)) = 0.1 \cdot 1.0 \cdot 0.9 = 0.09$$
$$Pr_{s_0}(C(s_0 s_2 s_2 s_2)) = 0.9 \cdot 1.0 \cdot 1.0 = 0.90$$

6.3 Probabilistic computation tree logic

To specify some properties of interest for DTMCs, we use *probabilistic computation tree logic* (PCTL) [HJ94], a probabilistic extension of the temporal logic CTL.

Definition 6.10 *The syntax of PCTL is defined as follows:*

$$\begin{array}{rcl} \Phi & ::= & true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathsf{P}_{\bowtie p}[\phi] \\ \phi & ::= & \mathsf{X}\ \Phi \mid \Phi\ \mathsf{U}^{\leq k}\ \Phi \end{array}$$

where a *is an atomic proposition,* $\bowtie \in \{<, \leq, \geq, >\}$*,* $p \in [0,1]$ *and* $k \in \mathbb{N} \cup \{\infty\}$*.*

In the above definition, we distinguish between state formulae Φ and path formulae ϕ, which are evaluated over states and paths, respectively. State formulae are self-explanatory, except for $\mathsf{P}_{\bowtie p}[\phi]$. It is satisfied by state s if the probability of taking a path from s satisfying ϕ is in the interval specified by $\bowtie p$. For the two path formulae, $\mathsf{X}\ \Phi$ is true if Φ is satisfied in the next state and $\Phi\ \mathsf{U}^{\leq k}\ \Psi$ is true if Ψ is satisfied within k time-steps and Φ is true up until that point.

The semantics of PCTL over DTMCs is defined as follows.

Definition 6.11 *Let $M = (S, s^\circ, \boldsymbol{P}, L)$ be a DTMC. For any state $s \in S$, the satisfaction relation $\models$ is defined inductively by:*

$$\begin{array}{rcl} s \models true & for\ all\ s \in S & \\ s \models a & \Leftrightarrow & a \in L(s) \\ s \models \neg\Phi & \Leftrightarrow & s \not\models \Phi \\ s \models \Phi \wedge \Psi & \Leftrightarrow & s \models \Phi \wedge s \models \Psi \\ s \models \mathsf{P}_{\bowtie p}[\phi] & \Leftrightarrow & Prob^M(s, \phi) \bowtie p \end{array}$$

where

$$Prob^M(s, \phi) := Pr_s\{\xi \in Path^M(s) \mid \xi \models \phi\}$$

and for any path $\xi \in Path^M(s)$:

$$\begin{array}{rcl} \xi \models \mathsf{X}\ \Phi & \Leftrightarrow & \xi(1) \models \Phi \\ \xi \models \Phi\ \mathsf{U}^{\leq k}\ \Psi & \Leftrightarrow & \exists i \in \mathbb{N} : i \leq k \wedge \xi(i) \models \Psi \wedge (\forall j < i : \xi(j) \models \Phi). \end{array}$$

Note that, for any state and path formula ϕ, the set $\{\xi \in Path^M(s) \mid \xi \models \phi\}$ is a measurable set of $(Path^M(s), \mathcal{X}_{Path^M(s)})$, see for example [Var85a], and hence Pr_s is well defined over this set. Observe that some useful operators are derivable, as shown by the following logical equivalences:

$$\begin{array}{rcl} false & \equiv & \neg true \\ \Phi \vee \Psi & \equiv & \neg(\neg\Phi \wedge \neg\Psi) \\ \Phi \rightarrow \Psi & \equiv & \neg\Phi \vee \Psi \\ \Diamond\Phi & \equiv & true\ \mathsf{U}^{\leq\infty}\ \Phi \\ \Diamond^{\leq k}\Phi & \equiv & true\ \mathsf{U}^{\leq k}\ \Phi \\ \Box\Phi & \equiv & \neg\Diamond\neg\Phi \\ \Box^{\leq k}\Phi & \equiv & \neg\Diamond^{\leq k}\neg\Phi \\ \exists\Diamond\Phi & \equiv & \mathsf{P}_{>0}[\Diamond\Phi] \\ \mathsf{P}_{\bowtie p}[\phi] & \equiv & \mathsf{P}_{\overline{\bowtie} 1-p}[\neg\phi] \end{array}$$

where $\overline{<} \equiv >$, $\overline{\leq} \equiv \geq$, $\overline{\geq} \equiv \leq$ and $\overline{>} \equiv <$. Intuitively, $\Diamond\Phi$ means that Φ is *eventually* satisfied and its bound variant $\Diamond^{\leq k}\Phi$ means that Φ is satisfied within k time-steps. Dually, a path satisfies $\Box\Phi$ if every state of the path satisfies Φ and its bound variant $\Box^{\leq k}\Phi$ means that Φ is true in the first k states of the path. Notice that $\forall\Diamond\Phi$ and $\mathsf{P}_{\geq 1}[\Diamond\Phi]$ are not equivalent. For example, consider the DTMC in Figure 6.1. State s_0 satisfies $\mathsf{P}_{\geq 1}[\Diamond succ]$ since the probability of reaching s_2 is 1. However, there is an infinite path $s_0 s_1 s_0 s_1 \ldots$ which never reaches s_2. Hence, $\forall\Diamond succ$ is not satisfied in state s_0.

6.4 Model checking PCTL

Given a DTMC $M = (S, s^\circ, \mathbf{P}, L)$ and a PCTL formula Φ, we are interested in the states which satisfy Φ, i.e. the set $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$. They can be computed by the model checking algorithm presented in [CY88, HJ94, CY95].

The algorithm has the same structure as that of the model checking algorithm for CTL [CES86]. Firstly, the parse tree of the formula Φ is constructed. Each node of the tree is labelled with a subformula of Φ; the root is labelled with Φ itself and the leaves are labelled with either *true* or an

atomic proposition. The algorithm recursively computes the set of states satisfying each subformula in a bottom-up way:

$$\begin{aligned} Sat(true) &= S \\ Sat(a) &= \{s \in S \mid a \in L(s)\} \\ Sat(\neg\Phi) &= S \backslash Sat(\Phi) \\ Sat(\Phi \wedge \Psi) &= Sat(\Phi) \cap Sat(\Psi) \\ Sat(\mathsf{P}_{\bowtie p}[\phi]) &= \{s \in S \mid Prob^M(s, \phi) \bowtie p\}. \end{aligned}$$

For most of the formulae, model checking is trivial to implement. The exceptions are formulae of the form $\mathsf{P}_{\bowtie p}[\phi]$. We distinguish two cases: $\mathsf{P}_{\bowtie p}[\mathsf{X}\ \Phi]$ and $\mathsf{P}_{\bowtie p}[\Phi\ \mathsf{U}^{\leq k}\ \Psi]$, and we assume that the sets $Sat(\Phi), Sat(\Psi)$ have already been computed.

$\mathsf{P}_{\bowtie p}[\mathsf{X}\ \Phi]$ **formulae** In this case, we need to compute the probability $Prob^M(s, \mathsf{X}\ \Phi)$ for each $s \in S$. For any particular state s, we have

$$Prob^M(s, \mathsf{X}\ \Phi) = \sum_{s' \in Sat(\Phi)} \mathbf{P}(s, s').$$

To obtain the vector $\underline{Prob^M}(\mathsf{X}\ \Phi)$ of probabilities for all states, we use a state-indexed column vector $\underline{\Phi}$ with $\underline{\Phi}(s) = 1$ if $s \in Sat(\Phi)$, 0 otherwise. Then $\underline{Prob^M}(\mathsf{X}\ \Phi)$ is calculated via the matrix-vector multiplication

$$\underline{Prob^M}(\mathsf{X}\ \Phi) = \mathbf{P} \cdot \underline{\Phi}\ .$$

$\mathsf{P}_{\bowtie p}[\Phi\ \mathsf{U}^{\leq k}\ \Psi]$ **formulae** In this cse, we need to compute the probabilities $Prob^M(s, \Phi\ \mathsf{U}^{\leq k}\ \Psi)$ for all state s where $k \in \mathbb{N} \cup \{\infty\}$.

Case 1: $k \in \mathbb{N}$. For $s \in S$ and $k \in \mathbb{N}$, it holds that

$$Prob^M(s, \Phi\ \mathsf{U}^{\leq k}\ \Psi) = \begin{cases} 1 & \text{if } s \in Sat(\Psi) \\ 0 & \text{if } (s \notin Sat(\Psi) \wedge k = 0) \vee s \in Sat(\neg\Phi \wedge \neg\Psi) \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob^M(s', \Phi\ \mathsf{U}^{\leq k-1}\ \Psi) & \text{otherwise.} \end{cases}$$

Definition 6.12 *Given a transition probability matrix $\boldsymbol{P}$, let the transformed matrix $\boldsymbol{P}[\Phi]$ be defined as follows: if $s \not\models \Phi$ then $\boldsymbol{P}[\Phi](s, s') = \boldsymbol{P}(s, s')$; if $s \models \Phi$ then $\boldsymbol{P}[\Phi](s, s) = 1$ and $\boldsymbol{P}[\Phi](s, s') = 0$ for all $s' \neq s$.*

It is shown in [KNP07] that the vector of probabilities $\underline{Prob^M(\Phi\ \mathsf{U}^{\leq k}\ \Psi)}$ can be computed using the following matrix and vector multiplications

$$\underline{Prob^M(\Phi\ \mathsf{U}^{\leq k}\ \Psi)} = (\mathbf{P}[\neg\Phi \vee \Psi])^k \cdot \underline{\Psi}\ .$$

Case 2: $k = \infty$. We abbreviate U^∞ as U. The vector of probabilities $\underline{Prob^M}(\Phi\ \mathsf{U}\ \Psi)$ can be computed as the least solution of the linear equation system

$$Prob^M(s, \Phi\ \mathsf{U}\ \Psi) = \begin{cases} 1 & \text{if } s \in Sat(\Psi) \\ 0 & \text{if } s \in Sat(\neg\Phi \wedge \neg\Psi) \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob^M(s', \Phi\ \mathsf{U}\ \Psi) & \text{otherwise.} \end{cases}$$

To simplify the computation we transform this equation system into one with a unique solution. We first find all the states s for which $Prob^M(s, \Phi\ \mathsf{U}\ \Psi)$ is exactly 0 or 1, i.e. we compute the two sets of states:

$$\begin{aligned} Sat(\mathsf{P}_{\leq 0}[\Phi\ \mathsf{U}\ \Psi]) &= \{s \in S \mid Prob^M(s, \Phi\ \mathsf{U}\ \Psi) = 0\} \\ Sat(\mathsf{P}_{\geq 1}[\Phi\ \mathsf{U}\ \Psi]) &= \{s \in S \mid Prob^M(s, \Phi\ \mathsf{U}\ \Psi) = 1\}. \end{aligned}$$

They can be determined with the algorithms **Prob0** and **Prob1**.

Procedure 7 Prob0$(Sat(\Phi), Sat(\Psi))$

$R := Sat(\Psi)$
$done := false$
while $done = false$ **do**
  $R' := R \cup \{s \in Sat(\Phi) \mid \exists s' \in R : \mathbf{P}(s, s') > 0\}$
  if $R' = R$ **then**
    $done := true$
  end if
  $R := R'$
end while
return $S \backslash R$

Procedure 8 Prob1$(Sat(\Phi), Sat(\Psi), Sat(\mathsf{P}_{\leq 0}[\Phi \ \mathsf{U} \ \Psi]))$

$R := Sat(\mathsf{P}_{\leq 0}[\Phi \ \mathsf{U} \ \Psi])$
$done := false$
while $done = false$ **do**
  $R' := R \cup \{s \in Sat(\Phi) \backslash Sat(\Psi) \mid \exists s' \in R : \mathbf{P}(s, s') > 0\}$
  if $R' = R$ **then**
    $done := true$
  end if
  $R := R'$
end while
return $S \backslash R$

The probabilities $Prob^M(s, \Phi \ \mathsf{U} \ \Psi)$ can then be computed as the unique solution of the following linear equation system

$$Prob^M(s, \Phi \ \mathsf{U} \ \Psi) = \begin{cases} 1 & \text{if } s \in Sat(\mathsf{P}_{\geq 1}[\Phi \ \mathsf{U} \ \Psi]) \\ 0 & \text{if } s \in Sat(\mathsf{P}_{\leq 0}[\Phi \ \mathsf{U} \ \Psi]) \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob^M(s', \Phi \ \mathsf{U} \ \Psi) & \text{otherwise.} \end{cases}$$

The linear equation system can be solved by many standard methods such as Gaussian elimination, L/U decomposition, Jacobi and Gauss-Seidel iteration.

Discrete-time Markov chains can be generalised to a model called *continuous-time Markov chains* (CTMC), mainly used in performance analysis. Both DTMCs and CTMCs can be extended with rewards. For a detailed account of these models and their use with the probabilistic model checker PRISM, see [KNP07].

6.5 Markov decision processes

Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals. Markov decision processes are obtained form probabilistic Kripke structures by adding rewards to each transition. The formal definition is as follows.

Definition 6.13 *A* Markov decision process (MDP) *is a tuple* $(S, s^\circ, Steps, \boldsymbol{r})$ *where*

- S *is a set of states;*
- s° *is the initial state;*
- $Steps : S \to 2^{\mathcal{D}(S)}$ *is a probabilistic transition function;*
- $\boldsymbol{r} : S \times \mathcal{D}(S) \to \mathbb{R}_{\geq 0}$ *is a reward function.*

A probabilistic transition $s \xrightarrow{\Delta} s'$ is made from a state s by first nondeterministically selecting a distribution $\Delta \in \mathit{Steps}(s)$ and then making a probabilistic choice of target state s' according to the distribution Δ. A path of an MDP represents a particular resolution of both nondeterminism and probability. Formally, a path of an MDP is a non-empty finite or infinite sequence of probabilistic transitions:

$$\xi = s_0 \xrightarrow{\Delta_0} s_1 \xrightarrow{\Delta_1} \cdots$$

such that $\Delta_i(s_{i+1}) > 0$ for all $i \geq 0$. We denote by $\xi(i)$ the state s_i and by $\xi[i]$ the distribution Δ_i. For a finite path ξ, we let $|\xi|$ denote the length of ξ (i.e. the number of transitions) and $\mathit{last}(\xi)$ be its final state. Finally, $\xi^{(i)}$ denotes the prefix of length i of ξ.

In contrast to a path, an adversary (sometimes also known as a *scheduler* or *policy*) represents a particular resolution of nondeterminism *only*. More precisely, an adversary is a function mapping every finite path ξ to a distribution $\Delta \in \mathit{Steps}(\mathit{last}(\xi))$. For any state $s \in S$ and adversary A, let $\mathit{Path}^A_f(s)$ and $\mathit{Path}^A(s)$ denote the sets of finite and infinite paths starting in s that correspond to A.

Definition 6.14 *An adversary A is called* memoryless *(or* simple*) if, for any finite paths ξ and ξ' for which $\mathit{last}(\xi) = \mathit{last}(\xi')$, we have $A(\xi) = A(\xi')$.*

The behaviour of an MDP from a state s, under a given adversary A, is fully probabilistic and is described by a probability space $(\mathit{Path}^A(s), \mathcal{X}_{\mathit{Path}^A(s)}, \mathit{Pr}^A_s)$ over the infinite paths corresponding to A that start in s. This can be defined in standard fashion, as we did in Section 6.2. Based on this, we introduce two quantitative measures for MDPs which together form the basis for probabilistic model checking of MDPs [dA97, BK98]. The first measure is *probabilistic reachability*, which refers to the probability of reaching a set of target states. For adversary A, the probability of reaching the target set $F \subseteq S$ from state s is given by:

$$p^A_s(F) := \mathit{Pr}^A_s\{\xi \in \mathit{Path}^A(s) \mid \exists i \in \mathbb{N} : \xi(i) \in F\}.$$

The second measure is *expected reachability*, which refers to the expected reward accumulated before reaching a set of target states. For an adversary A, the expected reward of reaching the target set F from state s, denoted $e^A_s(F)$, is defined as the usual expectation of the function $r(F, \cdot)$ (which returns, for a given path, the total reward accumulated until a state in F is reached along the path) with respect to the probability measure Pr^A_s. More precisely:

$$e^A_s(F) := \int_{\xi \in \mathit{Path}^A(s)} r(F, \xi)\ \mathrm{d}\mathit{Pr}^A_s$$

where for any path $\xi \in \mathit{Path}^A(s)$:

$$r(F, \xi) := \begin{cases} \sum_{i=1}^{\min\{j \mid \xi(j) \in F\}} \mathbf{r}(\xi(i-1), \xi[i-1]) & \text{if } \exists j \in \mathbb{N} : \xi(j) \in F \\ \infty & \text{otherwise.} \end{cases}$$

For simplicity, we have defined the reward of a path which does not reach F to be 1, even though the total reward of the path may not be infinite. Essentially, this means that the expected reward of reaching F from s under A is finite if and only if, under the adversary A, a state in F is reached from s with probability 1. Quantifying over all adversaries, we consider both the minimum and maximum values of these measures.

Definition 6.15 *For an MDP $(S, s^\circ, \mathit{Steps}, \boldsymbol{r})$, reachability objective $F \subseteq S$ and state $s \in S$, the* minimum *and* maximum reachability probabilities *of reaching F from s are defined by*

$$p^{\min}_s(F) = \inf_A p^A_s(F) \text{ and } p^{\max}_s(F) = \sup_A p^A_s(F)$$

and the minimum *and* maximum expected rewards *of reaching F from s are defined by*

$$e^{\min}_s(F) = \inf_A e^A_s(F) \text{ and } e^{\max}_s(F) = \sup_A e^A_s(F).$$

Computing values for probabilistic and expected reachability reduces to the *stochastic shortest path problem* for Markov decision processes; see for example [BT91, dA99]. A key result is that optimality with respect to probabilistic and expected reachability can always be achieved with memoryless adversaries. A consequence of this is that these quantities can be computed through an iterative processes known as value iteration, as the following proposition says.

Proposition 6.16 *Consider an MDP $(S, s^\circ, Steps, \boldsymbol{r})$ and set of target states $F \subseteq S$.*

- *The sequences of vectors $\langle p_n^{\min} \rangle_{n\in\mathbb{N}}$ and $\langle p_n^{\max} \rangle_{n\in\mathbb{N}}$ converge to the minimum and maximum probability of reaching the target set F, where for any state $s \in S$:*
  - *if $s \in F$, then $p_n^{\min}(s) = p_n^{\max}(s) = 1$ for all $n \in \mathbb{N}$;*
  - *if $s \notin F$, then:*

$$\begin{array}{rcl} p_n^{\min}(s) & = & \begin{cases} 0 & \text{if } n = 0 \\ \min_{\Delta\in Steps(s)} \sum_{s'\in S} \Delta(s') \cdot p_{n-1}^{\min}(s') & \text{otherwise} \end{cases} \\ p_n^{\max}(s) & = & \begin{cases} 0 & \text{if } n = 0 \\ \max_{\Delta\in Steps(s)} \sum_{s'\in S} \Delta(s') \cdot p_{n-1}^{\max}(s') & \text{otherwise.} \end{cases} \end{array}$$

- *The sequences of vectors $\langle e_n^{\min} \rangle_{n\in\mathbb{N}}$ and $\langle e_n^{\max} \rangle_{n\in\mathbb{N}}$ converge to the minimum and maximum expected reward of reaching the target set F, where for any state $s \in S$:*
  - *if $s \in F$, then $e_n^{\min}(s) = e_n^{\max}(s) = 1$ for all $n \in \mathbb{N}$;*
  - *if $s \notin F$, then:*

$$\begin{array}{rcl} e_n^{\min}(s) & = & \begin{cases} \infty & \text{if } p^{\max}(s) < 1 \\ 0 & \text{if } p^{\max}(s) = 1 \text{ and } n = 0 \\ \min_{\Delta\in Steps(s)}(\boldsymbol{r}(s,\Delta) + \sum_{s'\in S} \Delta(s') \cdot e_{n-1}^{\min}(s')) & \text{otherwise} \end{cases} \\ e_n^{\max}(s) & = & \begin{cases} \infty & \text{if } p^{\min}(s) < 1 \\ 0 & \text{if } p^{\min}(s) = 1 \text{ and } n = 0 \\ \max_{\Delta\in Steps(s)}(\boldsymbol{r}(s,\Delta) + \sum_{s'\in S} \Delta(s') \cdot e_{n-1}^{\max}(s')) & \text{otherwise.} \end{cases} \end{array}$$

□

6.6 Stochastic games

In this section, we review (simple) stochastic games [Sha53, Con92], which are turn-based games involving two players and chance.

Definition 6.17 *A* stochastic two-player game *is a tuple $\mathcal{G} = ((V, E), v^\circ, (V_1, V_2, V_p), \delta, \boldsymbol{r})$ where:*

- *(V, E) is a finite directed graph;*
- *$v^\circ \in V$ is an initial vertex;*
- *(V_1, V_2, V_p) is a partition of V;*
- *$\delta : V_p \to \mathcal{D}(V)$ is the probabilistic transition function;*
- *$\boldsymbol{r} : E \to \mathbb{R}_{\geq 0}$ is a reward function over edges.*

Vertices in V_1, V_2 and V_p are called "player 1", "player 2" and "probabilistic" vertices, respectively.

The game operates as follows. Initially, a token is placed on the starting vertex v°. At each step of the game, the token moves from its current vertex v to a neighbouring vertex v' in the game graph. The choice of v' depends on the type of the vertex v. If $v \in V_1$ then player 1 chooses v', if $v \in V_2$ then player 2 makes the choice, and if $v \in V_p$ then v' is selected randomly according to the distribution $\delta(v)$.

A Markov decision process can be thought of as a stochastic game in which there are no player 2 vertices and where there is a strict alternation between player 1 and probabilistic vertices.

A *play* in the game $\mathcal{G}$ is a sequence of vertices $\xi = v_0 v_1 v_2 ...$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. We denote by $\xi(i)$ the vertex v_i and, for a finite play ξ, we write $last(\xi)$ for the final vertex of ξ and $|\xi|$ for its length (the number of transitions). The prefix of length i of play ξ is denoted $\xi^{(i)}$.

A strategy for player 1 is a function $\sigma_1 : V^* V_1 \to \mathcal{D}(V)$, i.e. a function from the set of finite plays ending in a player 1 vertex to the set of distributions over vertices, such that for any $\xi \in V^* V_1$ and $v \in V$, if $\sigma_1(\xi)(v) > 0$, then $(last(\xi), v) \in E$. Strategies for player 2, denoted by σ_2, are defined analogously. For a fixed pair of strategies (σ_1, σ_2) we denote by $Play_f^{\sigma_1,\sigma_2}(v)$ and $Play^{\sigma_1,\sigma_2}(v)$ the set of finite and infinite plays starting in vertex v that correspond to these strategies. For strategy pair (σ_1, σ_2), the behaviour of the game is completely random and, for any vertex v, we can construct a probability space $(Play^{\sigma_1,\sigma_2(v)}, \mathcal{X}_{Play^{\sigma_1,\sigma_2(v)}}, Pr_v^{\sigma_1,\sigma_2})$. This construction proceeds similarly to MDPs (cf. Section 6.5).

A *reachability objective* of a game $\mathcal{G}$ is a set of vertices F which a player attempts to reach. For a fixed strategy pair (σ_1, σ_2) and vertex $v \in V$ we define both the probability and expected reward corresponding to the reachability objective F as:

$$p_v^{\sigma_1,\sigma_2}(F) := Pr_v^{\sigma_1,\sigma_2}\{\xi \in Play^{\sigma_1,\sigma_2(v)} \mid \exists i \in \mathbb{N} : \xi(i) \in F\}$$

$$e_v^{\sigma_1,\sigma_2}(F) := \int_{\xi \in Play^{\sigma_1,\sigma_2}(v)} r(F,\xi) \; \mathrm{d}Pr_v^{\sigma_1,\sigma_2}$$

where for any play $\xi \in Play^{\sigma_1,\sigma_2}(v)$:

$$r(F,\xi) := \begin{cases} \sum_{i=1}^{\min\{j|\xi(j)\in F\}} \mathbf{r}(\xi(i-1),\xi(i)) & \text{if } \exists j \in \mathbb{N} : \xi(j) \in F \\ \infty & \text{otherwise.} \end{cases}$$

Definition 6.18 *For a game $\mathcal{G} = ((V,E), v^\circ, (V_1, V_2, V_p), \delta, \boldsymbol{r})$, reachability objective $F \subseteq V$ and vertex $v \in V$, the optimal probabilities of the game for player 1 and player 2, with respect to F and v, are defined as follows:*

$$\sup_{\sigma_1}\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) \quad \textit{and} \quad \sup_{\sigma_2}\inf_{\sigma_1} p_v^{\sigma_1,\sigma_2}(F)$$

and the optimal expected rewards are:

$$\sup_{\sigma_1}\inf_{\sigma_2} e_v^{\sigma_1,\sigma_2}(F) \quad \textit{and} \quad \sup_{\sigma_2}\inf_{\sigma_1} e_v^{\sigma_1,\sigma_2}(F).$$

A player 1 strategy σ_1 is optimal from vertex v with respect to the probability of the objective if:

$$\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) \;=\; \sup_{\sigma_1}\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F).$$

The optimal strategies for player 2 and for expected rewards can be defined analogously.

Definition 6.19 *A strategy σ_i is* pure *if it does not use randomisation, that is, for any finite play ξ such that $last(\xi) \in V_i$, there exists $v' \in V$ such that $\sigma_i(\xi)(v') = 1$. A strategy σ_i is* memoryless *if its choice depends only on the current vertex, that is, $\sigma_i(\xi) = \sigma_i(\xi')$ for any finite plays ξ and ξ' such that $last(\xi) = last(\xi')$.*

Similarly to MDPs, for any stochastic game, the family of pure memoryless strategies suffices for optimality with respect to reachability objectives.

Proposition 6.20 *Consider a stochastic game $\mathcal{G} = ((V,E), v^\circ, (V_1, V_2, V_p), \delta, \boldsymbol{r})$ and set of target vertices $F \subseteq V$.*

- *The sequence of vectors $\langle p_n \rangle_{n\in\mathbb{N}}$ converges to the optimal probabilities for player 1 with respect to the reachability objective F, where for any vertex $v \in V$:*
  - *if $v \in F$, then $p_n(v) = 1$ for all $n \in \mathbb{N}$;*

– *and otherwise:*

$$p_n(v) = \begin{cases} 0 & \text{if } n = 0 \\ \max_{(v,v')\in E} p_{n-1}(v') & \text{if } n > 0 \text{ and } v \in V_1 \\ \min_{(v,v')\in E} p_{n-1}(v') & \text{if } n > 0 \text{ and } v \in V_2 \\ \sum_{v'\in V} \delta(v)(v') \cdot p_{n-1}(v') & \text{if } n > 0 \text{ and } v \in V_p. \end{cases}$$

- *The sequence of vectors $\langle e_n \rangle_{n\in\mathbb{N}}$ converges to the optimal expected rewards for player 1 with respect to the reachability objective F, where for any vertex $v \in V$:*
  - *if $v \in F$, then $e_n(v) = 0$ for all $n \in \mathbb{N}$;*
  - *if $\sup_{\sigma_2}\inf_{\sigma_1} p_v^{\sigma_1,\sigma_2}(F) < 1$, then $e_n(v) = \infty$ for all $n \in \mathbb{N}$;*
  - *and otherwise:*

$$e_n(v) = \begin{cases} 0 & \text{if } n = 0 \\ \max_{(v,v')\in E} (\boldsymbol{r}(v,v') + e_{n-1}(v')) & \text{if } n > 0 \text{ and } v \in V_1 \\ \min_{(v,v')\in E} (\boldsymbol{r}(v,v') + e_{n-1}(v')) & \text{if } n > 0 \text{ and } v \in V_2 \\ \sum_{v'\in V} (\boldsymbol{r}(v,v') + \delta(v)(v') \cdot e_{n-1}(v')) & \text{if } n > 0 \text{ and } v \in V_p. \end{cases}$$

□

The above proposition is a counterpart of Proposition 6.16 for MDP solutions; it forms the basis of an iterative method to compute the vector of optimal values for a game. Note that although this concerns only the optimal probability for player 1, similar results hold for player 2. See [Con92, Con93, CdAH04] for more details about results in this respect.

Bibliography

[AB01] S. Andova and J.C.M. Baeten. Abstraction in probabilistic process algebra. In Proceedings of the 7th International Conference on *Tools and Algorithms for the Construction and Analysis of Systems*, volume 2031 of Lecture Notes in Computer Science, pages 204–219. Springer, 2001.

[ABW06] S. Andova, J.C.M. Baeten, and T.A.C. Willemse. A complete axiomatisation of branching bisimulation for probabilistic systems with an application in protocol verification. In Proceedings of the 17th International Conference on *Concurrency Theory*, volume 4137 of Lecture Notes in Computer Science, pages 327–342. Springer, 2006.

[AJ94] S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic and Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.

[AW06] S. Andova and T.A.C. Willemse. Branching bisimulation for probabilistic systems: Characteristics and decidability. *Theoretical Computer Science*, 356(3):325–355, 2006.

[Bas96] Twan Basten. Branching bisimilarity is an equivalence indeed! *Information Processing Letters*, 58(3):141–147, 1996.

[BEMC00] Christel Baier, Bettina Engelen, and Mila E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *Journal of Computer and System Sciences*, 60(1):187–231, 2000.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[Bil95] P. Billingsley. *Probability and Measure*. Wiley, 1995.

[BK98] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BS01] Emanuele Bandini and Roberto Segala. Axiomatizations for probabilistic bisimulation. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2001.

[BT91] D. Bertsekas and J. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operation Research*, 16(3):580–595, 1991.

[CCR+03a] D. Cazorla, F. Cuartero, V.V. Ruiz, F.L. Pelayo, and J.J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *Journal of Logic and Algebraic Programming*, 55:57–103, 2003.

[CCR+03b] D. Cazorla, F. Cuartero, V.V. Ruiz, F.L. Pelayo, and J.J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *Journal of Logic and Algebraic Programming*, 55(1-2):57–103, 2003.

[CdAH04] K. Chatterjee, L. de Alfaro, and T. Henzinger. Trading memory for randomness. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems*, pages 206–217. IEEE Computer Society Press, 2004.

[CDSY99] R. Cleaveland, Z. Dayar, S.A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[CES86] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logics. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CHM90] Joseph Cheriyan, Torben Hagerup, and Kurt Mehlhorn. Can a maximum flow be computed on $\wr(nm)$ time? In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 235–248. Springer, 1990.

[Chr90] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In Proceedings the 3rd International Conference on *Concurrency Theory*, volume 458 of Lecture Notes in Computer Science, pages 126–140. Springer, 1990.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[Con93] A. Condon. *On algorithms for simple stochastic games*, volume 13, pages 203–224. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1993.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics-based tool for the verification of concurrency systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, 1993.

[CS01] R. Cleaveland and O. Sokolsky. *Equivalence and Preorder Checking for Finite-State Systems*, chapter 12, pages 391–424. North-Holland, 2001.

[CSZ92] R. Cleaveland, S.A. Smolka, and A.E. Zwarico. Testing preorders for probabilistic processes. In Proceedings of the 19th International Colloquium on *Automata, Languages and Programming*, volume 623 of Lecture Notes in Computer Science, pages 708–719. Springer, 1992.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite state probabilistic programs. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 338–345. IEEE Computer Society Press, 1988.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[dA97] Luca de Alfaro. *Formal verification of probabilistic systems.* PhD thesis, Stanford University, 1997.

[dA99] Luca de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1999.

[Der70] C. Derman. *Finite State Markovian Decision Processes.* Academic Press, 1970.

[DJGP02] Josee Desharnais, Radha Jagadeesan, Vineet Gupta, and Prakash Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, pages 413–422. IEEE Computer Society, 2002.

[DJGP04] Josee Desharnais, Radha Jagadeesan, Vineet Gupta, and Prakash Panangaden. Metrics for labelled markov processes. *Theoretical Computer Science*, 318(3):323–354, 2004.

[DNH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[DP05a] Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. In Proceedings of the 8th International Conference on *Foundations of Software Science and Computation Structures*, volume 3441 of Lecture Notes in Computer Science, pages 110–124. Springer, 2005.

[DP05b] Yuxin Deng and Catuscia Palamidessi. Axiomatizations for probabilistic finite-state behaviors. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structures*, volume 3441 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2005.

[DP07] Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. *Theoretical Computer Science*, 373(1-2):92–114, 2007.

[DPP05] Y. Deng, C. Palamidessi, and J. Pang. Compositional reasoning for probabilistic finite-state behaviors. In *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, volume 3838 of Lecture Notes in Computer Science, pages 309–337. Springer, 2005.

[DvGH+07a] Y. Deng, R.J. van Glabbeek, M. Hennessy, C.C. Morgan, and C. Zhang. Remarks on testing probabilistic processes. *ENTCS*, 172:359–397, 2007.

[DvGH+07b] Yuxin Deng, Rob van Glabbeek, Matthew Hennessy, Carroll Morgan, and Chenyi Zhang. Characterising testing preorders for finite probabilistic processes. In *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 313–325. IEEE Computer Society, 2007.

[DvGMZ07] Y. Deng, R.J. van Glabbeek, C.C. Morgan, and C. Zhang. Scalar outcomes suffice for finitary probabilistic testing. In Proc. *ESOP'07*, volume 4421 of Lecture Notes in Computer Science, pages 363–368. Springer, 2007.

[Eve79] Shimon Even. *Graph Algorithms*. Computer Science Press, 1979.

[FY03] Yuxi Fu and Zhenrong Yang. Tau laws for pi calculus. *Theoretical Computer Science*, 308:55–130, 2003.

[GJS90] Alessandro Giacalone, Chi-Chang Jou, and Scott A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of IFIP TC2 Working Conference on Programming Concepts and Methods*, 1990.

[Gla93] R.J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves. In Proc. *CONCUR'93*, volume 715 of Lecture Notes in Computer Science, pages 66–81. Springer, 1993.

[GRN99] C. Gregorio-Rodríguez and M. Núñez. Denotational semantics for probabilistic refusal testing. *ENTCS*, 22:111–137, 1999.

[GSST90] R.J. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In Proceedings of the 5th Annual IEEE Symposium on *Logic in Computer Science*, pages 130–141. Computer Society Press, 1990.

[GW96] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[Hen82] M. Hennessy. Powerdomains and nondeterministic recursive definitions. In *Proceedings of the 5th International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 178–193. Springer, 1982.

[Hen88] Matthew Hennessy. *An Algebraic Theory of Processes*. MIT Press, 1988.

[HHK95] Monika R. Henzinger, Thomas A. Henzinger, and Peter W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[HJ90] Hans Hansson and Bengt Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 278–287. IEEE Computer Society Press, 1990.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Hoa85a] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[Hoa85b] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[HSM97] Jifeng He, K. Seidel, and A.K. McIver. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–192, 1997.

[JHSY94] B. Jonsson, C. Ho-Stuart, and Wang Yi. Testing and refinement for nondeterministic and probabilistic processes. In Proceedings of the 3rd International Symposium on *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of Lecture Notes in Computer Science, pages 418–430. Springer, 1994.

[JL91] B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In Proceedings of the 6th Annual IEEE Symposium on *Logic in Computer Science*, pages 266–277. Computer Society Press, 1991.

[JP89] C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In Proceedings of the 4th Annual IEEE Symposium on *Logic in Computer Science*, pages 186–195. Computer Society Press, 1989.

[JS90] C.-C. Jou and S.A. Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In Proc. *CONCUR '90*, volume 458 of Lecture Notes in Computer Science, pages 367–383. Springer, 1990.

[JY95] B. Jonsson and Wang Yi. Compositional testing preorders for probabilistic processes. In Proceedings of the 10th Annual IEEE Symposium on *Logic in Computer Science*, pages 431–441. Computer Society Press, 1995.

[JY99] B. Jonsson and Wang Yi. Fully abstract characterization of probabilistic may testing. In Proceedings of the 5th International AMAST Workshop on *Formal Methods for Real-Time and Probabilistic Systems*, volume 1601 of Lecture Notes in Computer Science, pages 1–18. Springer, 1999.

[JY02] B. Jonsson and Wang Yi. Testing preorders for probabilistic processes can be characterized by simulations. *Theoretical Computer Science*, 282(1):33–51, 2002.

[JYL01] B. Jonsson, Wang Yi, and K.G. Larsen. Probabilistic extensions of process algebras. In *Handbook of Process Algebra*, chapter 11, pages 685–710. Elsevier, 2001.

[Kan42] L. Kantorovich. On the transfer of masses (in Russian). *Doklady Akademii Nauk*, 37(2):227–229, 1942.

[KKNP08] Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman, and David Parker. A game-based abstraction-refinement framework for markov decision processes. Technical Report RR-08-06, Computing Laboratory, Oxford University, 2008.

[KN98] M.Z. Kwiatkowska and G. Norman. A testing equivalence for reactive probabilistic processes. *Electronic Notes in Theoretical Computer Science*, 16(2), 1998.

[KNP07] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *Proceedings of the 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, volume 4486 of *Lecture Notes in Computer Science*, pages 220–270. Springer, 2007.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KS90] P.C. Kanellakis and S.A. Smolka. Ccs expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–65, 1990.

[Lin98] Huimin Lin. "on-the-fly" instantiation of value-passing processes. In *Proceedings of FORTE'98*, volume 135 of *IFIP Conference Proceedings*, pages 215–230. Kluwer, 1998.

[Low93] G. Lowe. Representing nondeterminism and probabilistic behaviour in reactive processes. Technical Report TR-11-93, Computing laboratory, Oxford University, 1993.

[LS91] Kim G. Larsen and Aren Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[LSV03] N. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. In Proceedings of the 14th International Conference on *Concurrency Theory*, volume 2761 of Lecture Notes in Computer Science, pages 204–222. Springer, 2003.

[Mar76] G. Markowsky. Chain-complete p.o. sets and directed sets with applications. *Algebra Universalis*, 6:53–68, 1976.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[Mil89a] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mil89b] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil91] Robin Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, Department of Computer Science, University of Edingburgh, 1991.

[Mil99] Robin Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[Mis00] M.W. Mislove. Nondeterminism and probabilistic choice: Obeying the laws. In Proc. of *CONCUR'00*, volume 1877 of Lecture Notes in Computer Science, pages 350–364. Springer, 2000.

[MMSS96] C.C. Morgan, A.K. McIver, K. Seidel, and J.W. Sanders. Refinement-oriented probability for CSP. *Formal Aspects of Computing*, 8(6):617–47, 1996.

[MO98] Markus Müller-Olm. Derivation of characteristic formulae. *Electronic Notes in Theoretical Computer Science*, 18:159–170, 1998.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Information and Computation*, 100:1–77, 1992.

[Núñ03] M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56:117–177, 2003.

[OH86] E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In Proceedings of 5th *GI Conference*, volume 104 of Lecture Notes in Computer Science, pages 167–183. Springer, 1981.

[PLS00] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak bisimulation for probabilistic systems. In *Proceedings of the 11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 334–349. Springer, 2000.

[PS96] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

[Rab63] M.O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[Ros94] W.R. Roscoe. *Model-checking CSP*. Prentice-Hall, 1994.

[Seg95] Roberto Segala. Modeling and verification of randomized distributed real-time systems. Technical Report MIT/LCS/TR-676, PhD thesis, MIT, Dept. of EECS, 1995.

[Seg96] R. Segala. Testing probabilistic automata. In *Proceedings of the 7th International Conference on Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 1996.

[Sei95] K. Seidel. Probabilistic communicating processes. *Theoretical Computer Science*, 152(2):219–249, 1995.

[Sha53] L. Shapley. Stochastic games. In *Proc. National Academy of Science*, volume 39, pages 1095–1100, 1953.

[SI94] Bernhard Steffen and Anna Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110:149–163, 1994.

[SL94] Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In *Proceedings of the 5th International Conference on Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 481–496. Springer, 1994.

[Sti97] Colin Stirling. Bisimulation, model checking and other games, 1997.

[SV03] M.I.A. Stoelinga and F.W. Vaandrager. A testing scenario for probabilistic automata. In Proceedings of the 30th International Colloquium on *Automata, Languages and Programming*, volume 2719 of Lecture Notes in Computer Science, pages 407–18. Springer, 2003.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[TKP05] R. Tix, K. Keimel, and G.D. Plotkin. Semantic domains for combining probability and non-determinism. *Electronic Notes in Theoretical Computer Science*, 129:1–104, 2005.

[Var85a] M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 327–338. IEEE Computer Society Press, 1985.

[Var85b] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In Proceedings 26th Annual Symposium on *Foundations of Computer Science*, pages 327–338, 1985.

[vBW01a] Franck van Breugel and James Worrell. An algorithm for quantitative verification of probabilistic transition systems. In *Proceedings of the 12th International Conference on Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2001.

[vBW01b] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic transition systems. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2001.

[vBW05] Franck van Breugel and James Worrell. A behavioural pseudometric for probabilistic transition systems. *Theoretical Computer Science*, 331(1):115–142, 2005.

[vG01] Rob van Glabbeek. The linear time - branching time spectrum I. In *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.

[vGW96] Rob J. van Gabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.

[YL92] Wang Yi and K.G. Larsen. Testing probabilistic and nondeterministic processes. In *Proceedings of the IFIP TC6/WG6.1 Twelfth International Symposium on Protocol Specification, Testing and Verification*, volume C-8 of *IFIP Transactions*, pages 47–61. North-Holland, 1992.

Index